รายงานผลโครงการวิจัย

แบบจำลองอนุภาคในบ่อศักย์แบบต่าง ๆ ในกลศาสตร์ควอนตัม

Simulation of Particles in Various Potential Wells in Quantum Mechanics

โดย

รองศาสตราจารย์วัชระ รอดสัมฤทธิ์

นางสาวอันชัญ หมวกงาม

นายเดี่ยว อภัยราช

สาขาวิชาฟิสิกส์ คณะวิทยาศาสตร์และเทคโนโลยี

มหาวิทยาลัยเทคโนโลยีราชมงคลธัญบุรี

(ได้รับเงินงบประมาณประจำปี 2553 หมวดเงินอุดหนุน)

บทคัดย่อ

โปรแกรมนี้ใช้ในการจำลองอนุภาคที่ถูกกักอยู่ในบ่อศักย์แบบต่าง ๆ โดยใช้การคำนวณเชิงตัวเลขกับสมการชเรอดิงเงอร์แบบ 1 มิติ บ่อศักย์ที่ใช้ในโปรแกรม ได้แก่บ่อศักย์แบบสี่เหลี่ยมที่มีความลึกจำกัด บ่อศักย์แบบพาราโบลา บ่อศักย์แบบสามเหลี่ยมสมมาตร บ่อศักย์แบบกำลังสองของแทนเจนต์ และบ่อศักย์กำลังสองของไฮเปอร์บอลิกโคไซน์ โปรแกรมได้แสดงค่าพลังงานของอนุภาคและฟังก์ชันคลื่นที่สมนัยกับพลังงานค่านั้น ๆ โดยแสดงผลเป็นกราฟิกบนจอภาพ รวมทั้งพลังงานและฟังก์ชันคลื่นที่เป็นค่าเจาะจงด้วย

เมื่อนำค่าพลังงานที่ได้จากการคำนวณเชิงตัวเลขของโปรแกรมเปรียบเทียบกับค่าที่ได้จากการวิเคราะห์พบว่า บ่อศักย์แบบสี่เหลี่ยมที่มีความลึกจำกัดเกิดความคลาดเคลื่อนในการคำนวณไม่เกิน 3% และบ่อศักย์แบบพาราโบลา ได้ผลการคำนวณตรงกับค่าที่ได้จากการวิเคราะห์ทุกประการ

Abstract

This computer program simulates and solves numerically the time independent one dimensional Schrodinger equation for a particle bounding to various finite potential well. Finite square well, parabolic well, symmetric triangular well, square tangent well and square hyperbolic cosine well are selected to investigated. The energy values and corresponding wave function of each potential well including eigen values are presented graphically.

The comparison with analytic method in finite square well and parabolic well provided the reasonable accuracy for numerical method in program simulation.

สารบัญ

ใช้งานโปรแกรม หรือ download Source code ล่าสุด ได้ที่

http://www.electron.rmutphysics.com/SE1D

บทที่ 1
บทนำ

1.1 ที่มาของปัญหา

สมการชเรอดิงเงอร์ (Schrödinger equation) เป็นสมการที่ใช้อธิบายพฤติกรรมของอนุภาคระดับจุลภาค ในวิชากลศาสตร์ควอนตัม ถ้าการเปลี่ยนแปลงของระบบควอนตัมไม่ขึ้นกับเวลา (time independent) สมการชเรอดิงเงอร์จะเป็นสมการอนุพันธ์ย่อยอันดับสอง มีรูปสมการดังนี้

$$\left(-\frac{\bar{h}^2}{2m}\nabla^2 + V(r)\right)\Psi(r) = E\Psi(r)$$

เมื่อ $\Psi(r)$ คือ ฟังก์ชันคลื่นของอนุภาคในระบบ

V (r) คือ พลังงานศักย์ในระบบนั้น

E คือ พลังงานอนุภาคในระบบที่สอดคล้องกับฟังก์ชันคลื่นนั้น

การหาคำตอบของสมการนี้ค่อนข้างยุ่งยากและซับซ้อน และขึ้นอยู่กับรูปแบบของพลังงานศักย์ตรงบริเวณที่อนุภาคนั้นอยู่ ถ้าอนุภาคถูกจำกัดให้เคลื่อนที่ในบริเวณที่มีพลังงานศักย์ที่มีค่าไม่เป็นศูนย์ ในเบื้องต้นจะต้องหาฟังก์ชันคลื่นของอนุภาคเพื่อที่จะอธิบายพฤติกรรมของมัน ตัวอย่างที่กล่าวถึงในหนังสือกลศาสตร์ควอนตัมทั่วไป ส่วนใหญ่จะยกตัวอย่างที่พลังงานศักย์อยู่ในรูปง่าย ๆ เช่น เป็นบ่อพลังงานศักย์แบบสี่เหลี่ยม (Square well potential) ถึงแม้จะจำกัดมิติการเคลื่อนที่ของอนุภาคให้เป็นแบบ 1 มิติ นักศึกษาก็ยังมองภาพการหาฟังก์ชันคลื่น และพลังงานซึ่งเป็นค่าเจาะจง (eigen value) ที่สอดคล้องกับฟังก์ชันคลื่นไม่ชัดเจน ทำให้การเรียนรู้วิธีการหาคำตอบของสมการชเรอดิงเงอร์ เป็นเรื่องที่เป็นนามธรรม เข้าใจยากและน่าเบื่อ

เพื่อที่จะแก้ปัญหาตรงจุดนี้ จึงจัดทำโปรแกรมจำลองการเคลื่อนที่ของอนุภาคในบ่อศักย์แบบ 1 มิติ สามารถทำงานในคอมพิวเตอร์แบบ PC ใช้การคำนวณเชิงตัวเลข หาคำตอบของสมการชเรอดิงเงอร์ของอนุภาคในบ่อศักย์ที่มีรูปแบบต่าง ๆ กัน พร้อมแสดงผลออกมาในรูปของกราฟิก สามารถกำหนดค่าพลังงานของอนุภาค แล้วให้คอมพิวเตอร์แสดงฟังก์ชันคลื่นของอนุภาคที่ค่าพลังงานนั้นออกมาให้เห็นอย่างทันทีทันใด ผู้เรียนสามารถเปรียบเทียบผลลัพธ์ที่ได้กับคำตอบที่ได้จากวิธีวิเคราะห์ทางคณิตศาสตร์ ทำให้การเรียนรู้กลศาสตร์ควอนตัมเป็นเรื่องที่น่าสนใจ และใช้เวลาในการเรียนรู้สั้นลง การดัดแปลงหรือเพิ่มเติมโปรแกรมอีกเพียงเล็กน้อย สามารถนำไปใช้อธิบายพฤติกรรมของอนุภาคซึ่งถูกกักไว้ในบ่อศักย์แบบอื่น ๆ ที่ไม่ได้กล่าวถึงในที่นี้ได้อีกด้วย

1.2. ความมุ่งหมายของการวิจัย

เพื่อพัฒนาโปรแกรม แสดงแบบจำลองอนุภาคในบ่อศักย์แบบต่าง ๆ สำหรับประกอบการเรียนการสอนวิชา กลศาสตร์ควอนตัม และวิชาฟิสิกส์ 2 (หัวข้อ กลศาสตร์ควอนตัมเบื้องต้น)

1.3 ประโยชน์ที่คาดว่าจะได้รับ

1. ได้โปรแกรมสำหรับใช้ในการจำลองอนุภาคในบ่อศักย์แบบต่าง ๆ ในกลศาสตร์ควอนตัม

2. สามารถนำไปประกอบบทเรียน E-learning เรียนรู้ผ่านเครือข่ายอินเตอร์เน็ตได้

1.4 ขอบเขตของโครงการวิจัย

แบบจำลองอนุภาคในบ่อศักย์ แบบต่าง ๆ ได้กำหนดขอบเขตของลักษณะของบ่อศักย์ที่ใช้ในแบบจำลอง ดังต่อไปนี้

1. บ่อศักย์ที่เป็นรูปสี่เหลี่ยม (Square well) สามารถเปลี่ยนระดับความลึกของบ่อได้จนถึงค่าอนันต์ โดยให้ความกว้างของบ่อศักย์คงที่

2. บ่อศักย์แบบพาราโบลา (Parabolic well) เป็นบ่อศักย์ที่เกิดจากฮาร์มอนิกออสซิลเลเตอร์ (Harmonic oscillator)

3. บ่อศักย์แบบสามเหลี่ยมสมมาตร (Symmetric triangular well) เป็นบ่อศักย์ที่เป็นรูปสามเหลี่ยมที่สมมาตรในแนวแกน y

4. บ่อศักย์แบบแทนเจนต์กำลังสอง (Square tangent Well) เป็นบ่อศักย์ที่เกิดจากค่ากำลังสองของแทนเจนต์ จะมีลักษณะคล้ายกับบ่อศักย์รูปสี่เหลี่ยมในข้อ 1. แต่ก้นหลุมจะมีพลังงานศักย์เป็นศูนย์

5. บ่อศักย์แบบไฮเปอร์บอลิกโคไซน์กำลังสอง (Square hyperbolic cosine Well) เป็นบ่อศักย์ที่เกิดจากค่ากำลังสองของ cosh x จะมีลักษณะคล้ายกับบ่อศักย์แบบพาราโบลาในข้อ 2. แต่ก้นหลุมของพลังงานศักย์จะมีค่าเป็นลบ

1.5 นิยามศัพท์เฉพาะ

กลศาสตร์ควอนตัม (Quantum mechanics) เป็นสาขาหนึ่งในวิชาฟิสิกส์ กล่าวถึงพฤติกรรมของอนุภาคขนาดเล็กระดับอะตอมหรือเล็กกว่าอะตอม พฤติกรรมของอนุภาคเหล่านี้จะไม่เป็นไปตามกฎของนิวตันในกลศาสตร์คลาสสิก (Classical mechanics) อนุภาคสามารถแสดงสมบัติความเป็นคลื่นได้ ในขณะเดียวกันคลื่นที่เคยเชื่อกันว่าต่างจากอนุภาคอย่างสิ้นเชิง ก็มีสมบัติความเป็นอนุภาคได้เช่นกัน ปริมาณทางฟิสิกส์หลาย ๆ ปริมาณที่เราเชื่อกันว่ามีค่าต่อเนื่อง กลับ

พบว่ามันมีค่าได้เพียงบางค่าเท่านั้น เช่น พลังงาน โมเมนตัมเชิงมุม การที่ปริมาณทางฟิสิกส์มีค่าไม่ต่อเนื่องหรือมีค่าเป็นช่วง ๆ เรียกลักษณะเช่นนี้ว่า Quantization กลศาสตร์ควอนตัมทำให้มุมมองของนักวิทยาศาสตร์ที่มีต่อปรากฏการณ์ในระดับจุลภาคแตกต่างไปจากเดิม สามารถตอบปัญหาต่าง ๆ ที่ฟิสิกส์ยุคเก่าไม่สามารถตอบได้ หรืออธิบายไม่ถูกต้อง นำไปสู่การค้นพบและประดิษฐ์คิดค้นสิ่งใหม่ ๆ เช่น สารกึ่งตัวนำ เลเซอร์ และนาโนเทคโนโลยี

บ่อศักย์ (Potential well) หมายถึงบริเวณที่มีพลังงานศักย์ซึ่งสามารถกักกันอนุภาคไว้ในบริเวณนั้น ๆ ในกลศาสตร์นิวตัน ถ้าอนุภาคมีพลังงานไม่มากพอที่จะหลุดพ้นจากสนามที่ทำให้เกิดพลังงานศักย์นั้น อนุภาคจะถูกกักขังไว้ในบริเวณนั้น ไม่สามารถหลุดพ้นไปจากบ่อนี้ได้พลังงานของอนุภาคในบ่อศักย์นั้นสามารถมีได้ทุก ๆ ค่า (แต่ต้องไม่เกินค่าที่จะทำให้อนุภาคนั้นหลุดพ้นไปจากบ่อ) แต่ในกลศาสตร์ควอนตัม พบว่าพลังงานของอนุภาคมีได้เพียงบางค่าเท่านั้น พลังงานของอนุภาคนี้ต้องเป็นค่าเจาะจงที่สอดคล้องกับฟังก์ชันคลื่นที่เป็น eigen function ของค่าพลังงานนี้ด้วย

ฟังก์ชันเจาะจง (Eigen function) หมายถึงฟังก์ชันใด ๆ เมื่อนำตัวปฏิบัติการทางคณิตศาสตร์กระทำกับฟังก์ชันนี้แล้ว ผลลัพธ์ที่ได้มาจะเป็นฟังก์ชันอันเดิมคูณกับจำนวนจริงค่าหนึ่ง เรียกจำนวนจริงที่ได้นี้ว่า เป็นค่าเจาะจง (Eigenvalue) ตัวอย่างเช่น f(x) = ce^{-2x} เมื่อใช้ตัวปฏิบัติการ $\frac{d}{dx}$ กระทำกับฟังก์ชัน f(x) นี้จะผลลัพธ์เป็น

$$\frac{df(x)}{dx} = \frac{d}{dx}(ce^{-2x}) = -2ce^{-2x}$$

หรือ $$\frac{df(x)}{dx} = -2f(x)$$

f(x) จัดเป็นฟังก์ชันเจาะจง เพราะเมื่อใช้การหาอนุพันธ์ $\frac{d}{dx}$ เป็นตัวปฏิบัติการกระทำกับ f(x) แล้วจะได้ผลลัพธ์เป็นฟังก์ชันเดิม ค่าเจาะจงที่ได้ในที่นี้คือ –2

ในสมการชเรอดิงเงอร์ นั้น $\Psi(r)$ เป็นฟังก์ชันเจาะจง โดยมี $\left(-\frac{\overline{h}^2}{2m}\nabla^2 + V(r)\right)$ เป็นตัวปฏิบัติการ และ E เป็นค่าเจาะจง

This page left blank intentionally

บทที่ 2
ทฤษฎีและเอกสารที่เกี่ยวข้อง

ในบทนี้กล่าวถึงทฤษฎีและเอกสารที่เกี่ยวข้องตามหัวข้อต่อไปนี้

2.1 สมการชเรอดิงเงอร์ 1 มิติ ที่ไม่ขึ้นกับเวลา

2.2 บ่อศักย์แบบต่าง ๆ

2.3 วิธีการคำนวณเชิงตัวเลขที่ใช้ในงานวิจัย

2.4 งานวิจัยหรือโปรแกรมเกี่ยวกับการจำลองปรากฏการณ์ในกลศาสตร์ควอนตัม

2.1 สมการชเรอดิงเงอร์ 1 มิติ ที่ไม่ขึ้นกับเวลา

(Time independent Schrodinger equation in 1 dimension)

สมการชเรอดิงเงอร์ (Schrödinger equation) สำหรับระบบควอนตัมทั่วไป (General quantum system) มีรูปสมการดังนี้

$$\hat{H}\psi = i\bar{h}\frac{\partial\psi}{\partial t} \qquad (2.1)$$

เมื่อ Ψ คือ ฟังก์ชันคลื่นของอนุภาค

$i\bar{h}\frac{\partial}{\partial t}$ คือ ตัวปฏิบัติการพลังงาน (Energy operator)

$\hat{H}$ คือ ตัวปฏิบัติการแฮมิลโทเนียน(Hamiltonian operator)

กรณีเฉพาะสำหรับอนุภาคที่ถูกกักอยู่ในบ่อศักย์ โดยที่การเปลี่ยนแปลงของระบบควอนตัมนี้ไม่ขึ้นกับเวลา (time independent) สามารถเขียนสมการชเรอดิงเงอร์จากสมการ (2.1) ให้อยู่ในรูปของสมการอนุพันธ์ย่อยอันดับสอง คือ

$$\left(-\frac{\bar{h}^2}{2m}\nabla^2 + V(r)\right)\Psi(r) = E\Psi(r)$$

$$\hat{H} = -\frac{\bar{h}^2}{2m}\nabla^2 + V(r)$$

V (r) คือ พลังงานศักย์ที่ตำแหน่ง r ใด ๆ ในบริเวณนั้น ไม่ขึ้นกับเวลา

E คือ พลังงานทั้งหมดของระบบ

การหาผลเฉลยของสมการนี้เพื่อหาฟังก์ชันคลื่นของอนุภาคค่อนข้างยุ่งยากและซับซ้อน และขึ้นอยู่กับพลังงานศักย์ตรงบริเวณที่อนุภาคนั้นอยู่ด้วย ในที่นี้จึงใช้กรณีที่เข้าใจได้ง่ายคือ การเคลื่อนที่ของอนุภาคเป็นแบบ 1 มิติ ในบ่อศักย์ 1 มิติเช่นกัน สมการชเรอดิงเงอร์ จะกลายเป็น

$$\left(-\frac{\bar{h}^2}{2m}\frac{\partial^2}{\partial x^2}+V(x)\right)\Psi(x)=E\Psi(x)$$

$$-\frac{\bar{h}^2}{2m}\frac{\partial^2\psi(x)}{\partial x^2}+V(x)\psi(x)=E\psi(x)$$

$$\frac{\partial^2\psi(x)}{\partial x^2}+\frac{2m}{\bar{h}^2}\big(E-V(x)\big)\psi(x)=0 \qquad (2.2)$$

พิจารณาเทอม $\frac{2m}{\bar{h}^2}(E-V(x))$ หรือเขียนในรูป $\frac{2mc^2}{\bar{h}^2c^2}(E-V(x))$ ถ้าอนุภาคเป็นอิเล็กตรอน จะได้ $mc^2=5.11\times10^5 eV$ และ $\bar{h}c=197.3\ eV\ nm$ ปรากฏการณ์ที่มีขนาดเล็กระดับอะตอม นิยมใช้หน่วยในการวัดพลังงานเป็น eV และวัดขนาดความยาวเป็น nm

อย่างไรก็ตาม หน่วยในระบบ SI หรือ CGS เหมาะสมกับการใช้อธิบายปริมาณทางฟิสิกส์ในกลศาสตร์คลาสสิก แต่ถ้านำมาใช้ในกลศาสตร์ควอนตัม เช่นในการคำนวณเชิงตัวเลขที่เกี่ยวข้องกับค่าคงที่ของพลังค์ (Plank's constant) มีขนาดประมาณ 10^{-34} จูล.วินาที มวลของอิเล็กตรอนมีขนาดประมาณ 10^{-31} กิโลกรัม จะเกิดความคลาดเคลื่อนอันเกิดจากคอมพิวเตอร์คำนวณเลขจำนวนน้อย ๆ ได้ง่ายมาก และในกรณีที่มีการคำนวณซ้ำ ๆ หลายรอบ ความคลาดเคลื่อนนี้จะถูกสะสมไว้ ทำให้ผลลัพธ์ที่ได้ห่างไกลค่าแท้จริงอย่างสิ้นเชิง

เพื่อให้เกิดความสะดวกในการคำนวณ จึงเลือกใช้ หน่วยในระบบอะตอม หรือ Atomic Units หน่วยนี้จะกำหนดให้ค่าคงที่ของพลังค์ และมวลของอิเล็กตรอน มีขนาด 1 หน่วย ($m_e=\bar{h}=1$) การกำหนดเช่นนี้จึงเกิดหน่วยใหม่ขึ้นเป็นจำนวนมาก (ดูรายละเอียดในภาคผนวก 1) สมการ (2.2) จะถูกลดรูปเหลือเพียง

$$\frac{\partial^2\psi(x)}{\partial x^2}+2\big(E-V(x)\big)\psi(x)=0 \qquad (2.3)$$

สมบัติของฟังก์ชันคลื่นที่เป็นผลเฉลยของสมการชเรอดิงเงอร์ บางประการที่จะต้องนำไปใช้ในการคำนวณของโปรแกรม มีดังนี้

1. รูปแบบของฟังก์ชันจะเป็นฟังก์ชันคู่และฟังก์ชันคี่สลับกันไป (จะเป็นฟังก์ชันคู่หรือคี่นั้นโดยการเทียบกับเส้นที่แบ่งครึ่งตรงกลางของบ่อศักย์) และเป็นจริงเช่นนี้เสมอสำหรับพลังงานศักย์ทุกแบบ ที่มีคุณสมบัติสมมาตร หรือ V(–x) = V(x)

2. ฟังก์ชันคลื่นที่สอดคล้องกับจำนวนเต็ม n = 1, 2, 3, ... เรียก n ว่าเลขควอนตัม (quantum number) จะเป็นฟังก์ชันเจาะจง (eigen function) ใช้ในการอธิบายระดับพลังงานของ

อนุภาค (E, energy) ในบ่อศักย์ เรียก E ว่าเป็นค่าเจาะจง (eigenvalue) ที่สถานะ n =1 จะเป็นค่าพลังงานต่ำสุดของระบบ เรียกระดับพลังงานนี้ว่าเป็นระดับ ground state

3. ในการแปรความหมายฟังก์ชันคลื่น จะใช้ขนาดของฟังก์ชันคลื่น (magnitude of the wave function) เป็นตัวบ่งบอกถึงความน่าจะเป็นที่จะพบอนุภาคในบริเวณดังกล่าว เรียกว่า Probability density เขียนเป็นสัญลักษณ์คือ P(x,t)

$$P(x,t) = \psi^*(x,t)\psi(x,t) \qquad (2.4)$$

ซึ่งเป็นการหาอัมพลิจูดของฟังก์ชันคลื่นนั่นเอง ถ้ามีอนุภาคอยู่ในระบบหนึ่ง ความน่าจะเป็นที่จะพบอนุภาคในบริเวณนั้นควรมีค่าเป็น 1 หรือมีโอกาสที่จะพบอนุภาคนั้นอยู่ในบริเวณนั้นอย่างแน่นอน 100% นั่นคือ

$$\int_{-\infty}^{+\infty} \psi^*(x)\psi(x)dx = 1 \qquad (2.5)$$

สมการ (2.5) เป็นการปรับความสูงของอัมพลิจูดของฟังก์ชันคลื่น ขั้นตอนนี้เราเรียกว่า Normalization ฟังก์ชันคลื่น $\psi(x)$ และอนุพันธ์ของมันจะต้องมีค่าต่อเนื่องทุกค่าของ x จากสมการ (2.5) จะเห็นได้ว่า กำลังสองของฟังก์ชันคลื่นต้องสามารถหาปริพันธ์ได้ด้วย

4. ฟังก์ชันคลื่นของอนุภาคในบ่อศักย์ (ที่ไม่ใช่ตรงผนังของบ่อ) อาจมีค่าเป็นศูนย์ได้ในบางตำแหน่งของ x เราเรียกจุด x ที่ทำให้ฟังก์ชันคลื่น ณ ตำแหน่งนี้มีค่าเป็นศูนย์ว่า node หรือ zero crossing จำนวน node ของฟังก์ชันคลื่นของอนุภาคที่มีพลังงาน E_n พบว่ามีจำนวน n – 1 จุด และพบว่าที่ระดับพลังงานที่ n = 1 ไม่มี node เกิดขึ้นที่ระดับพลังงานนี้

5. ฟังก์ชันคลื่นของอนุภาคที่ถูก normalize แล้วมีคุณสมบัติเป็น orthogonal (ตั้งฉากซึ่งกันและกัน) หรือ orthonomal (เป็นทั้ง orthogonal และ normalize) สามารถเขียนเป็นสมการได้เป็น

$$\int_{-\infty}^{+\infty} \psi_m^*(x)\psi_n(x)dx = \delta_{mn}$$

เมื่อ δ_{mn} คือ Kronecker delta function มีค่าเท่ากับ 1 เมื่อ $m = n$ (ซึ่งก็คือสมการ (2.5) นั่นเอง) และมีค่าเป็น 0 เมื่อ $m \neq n$ คุณสมบัตินี้สอดคล้องกับคุณสมบัติของเวกเตอร์ เราจึงสามารถเขียนเซตของฟังก์ชันคลื่นในแบบ vector space ได้

6. สามารถเขียนฟังก์ชันคลื่นอยู่ในรูป complete set โดยที่ เราสามารถเขียนฟังก์ชัน f(x) ซึ่งเป็นฟังก์ชันใด ๆ ในรูปของฟังก์ชันคลื่นดังนี้

$$f(x) = \sum_{n=1}^{\infty} c_n \psi_n$$

การหาค่าสัมประสิทธิ์ c_n ซึ่งเป็นค่าคงที่ สามารถอาศัยสมบัติ Normalization และเทคนิคการแปลงแบบฟูเรียร์ (Fourier transformation) หาได้โดยตรงดังนี้

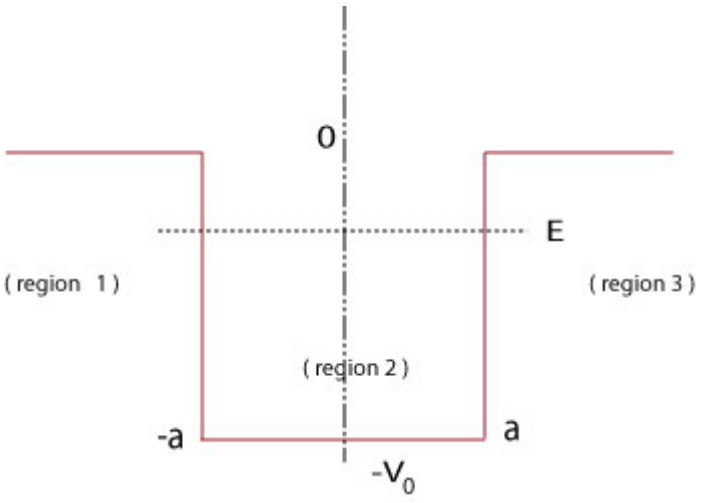
$$\int_{-\infty}^{\infty} \psi_n^*(x) f(x) dx = \sum_{n=1}^{\infty} c_n \int_{-\infty}^{\infty} \psi_n^*(x) \psi_n(x) dx = \sum_{n=1}^{\infty} c_n \delta_{mn}$$

$$\sum_{n=1}^{\infty} c_n = \int_{-\infty}^{\infty} \psi_n^*(x) f(x) dx$$

2.2 บ่อศักย์แบบต่าง ๆ

2.2.1 บ่อศักย์เป็นแบบสี่เหลี่ยมมีความลึกจำกัด (1 dimensional finite rectangular well) อนุภาคมวล m เคลื่อนที่ในบ่อศักย์ที่มีขอบเขต ดังนี้

$$V = \begin{cases} -V_0 & if \; -a < x < a \\ 0 & if \quad |x| \geq a \end{cases}$$



รูป 2.1 บ่อศักย์แบบสี่เหลี่ยมมีความลึกจำกัดใน 1 มิติ

อนุภาคที่มีพลังงานมากกว่าศูนย์ (E > 0) จะไม่ถูกกักไว้ในบ่อ แต่จะถูกกระเจิงออกไปโดยพลังงานศักย์ของบ่อ อนุภาคที่มีพลังงานน้อยกว่าศูนย์ ($-V_0 < E < 0$) จะถูกยึดไว้ในบ่อศักย์ เรียกกรณีนี้ว่าเป็น Bound state ตามทฤษฎีของกลศาสตร์คลาสสิก อนุภาคจะถูกกักและสะท้อนกลับไปกลับมาระหว่างขอบเขต $-a < x < a$ เท่านั้น โดยที่อนุภาคจะมีค่าพลังงานเป็นค่าใด ๆ ก็ได้

ในกลศาสตร์ควอนตัม พลังงานของอนุภาคจะมีค่าได้เพียงบางค่าเท่านั้น และค่านี้เป็นค่าเจาะจง (Eigen value) ของฟังก์ชันคลื่นที่เป็น Eigen function ที่สอดคล้องกัน

ในแต่ละบริเวณสามารถเขียน สมการ (2.2) ได้แตกต่างกันดังนี้

บริเวณที่ 1 (region 1) บริเวณนี้ $x < a$ แทนค่า V (x) = 0 E = –E

$$\frac{\partial^2\psi(x)}{\partial x^2} = \frac{2m}{\bar{h}^2} E\psi(x)$$

บริเวณที่ 2 (region 2) $-a \le x \le a$ แทนค่า V(x) = $-V_0$ และ E มีค่าติดลบ

$$\frac{\partial^2\psi(x)}{\partial x^2} + \frac{2m}{\bar{h}^2}(V_0 - E)\psi(x) = 0$$

บริเวณที่ 3 (region 3) $x > a$ แทนค่า V = 0

$$\frac{\partial^2\psi(x)}{\partial x^2} = \frac{2m}{\bar{h}^2} E\psi(x)$$

เพื่อสะดวกต่อการเขียนสมการ จึงกำหนดให้

$$\ell = \sqrt{\frac{2mE}{\bar{h}^2}} \qquad (2.6)$$

และ $$\kappa = \sqrt{\frac{2m}{\bar{h}^2}(V_0 - E)} \qquad (2.7)$$

ค่า ℓ และ κ เป็นจำนวนจริงบวก กรณี bound state $V_0 - E > 0$

บริเวณที่ 1 และบริเวณที่ 3 : V = 0 แทนค่า $\ell = \sqrt{\frac{2mE}{\bar{h}^2}}$ สมการจะกลายเป็น

$$\frac{\partial^2\psi}{\partial x^2} - \ell^2\psi = 0 \qquad (2.8)$$

บริเวณที่ 1 จะได้ผลเฉลยทั่วไป

$$\psi_1 = Ae^{\ell x} + Be^{-\ell x}$$

เมื่อ A และ B คือค่าคงที่ พิจารณาที่ $x \to -\infty$ จะเห็นว่าเทอม $Be^{-\ell x}$ มีค่ามาก นั่นหมายถึง ψ_1 มีค่ามากตามไปด้วย ซึ่งเป็นไปไม่ได้ ดังนั้น B จึงต้องมีค่าเป็นศูนย์ ผลเฉลยของสมการจะเหลือเพียง

$$\psi_1 = Ae^{\ell x}$$

บริเวณที่ 3 จะได้ผลเฉลยทั่วไปคือ

$$\psi_3 = Fe^{\ell x} + Ge^{-\ell x}$$

เมื่อ F และ G คือค่าคงที่ พิจารณาที่ $x \to +\infty$ จะเห็นว่าเทอม $Fe^{\ell x}$ มีค่ามาก อันที่จริง ψ_3 ควรมีค่าเป็นศูนย์ ดังนั้น F จึงต้องมีค่าเป็นศูนย์ ดังนั้นผลเฉลยของสมการคือ

$$\psi_3 = Ge^{-\ell x}$$

บริเวณที่ 2: V = –V(x) ค่าพลังงานของอนุภาค | E | มีค่าน้อยกว่า | V(x)| เมื่อแทนค่า $\kappa = \sqrt{\frac{2m}{\hbar^2}(V_0 - E)}$ แล้วรูปสมการจะได้เป็น

$$\frac{\partial^2 \psi}{\partial x^2} + \kappa^2 \psi = 0 \qquad (2.9)$$

ผลเฉลยทั่วไปของสมการนี้คือ

$$\psi_2 = Ce^{i\kappa x} + De^{-i\kappa x}$$

เมื่อ C และ D คือค่าคงที่

อาศัยเงื่อนไขขอบเขตตรงบริเวณรอยต่อ (Boundary condition) ที่ว่า ฟังก์ชันคลื่นและอนุพันธ์อันดับหนึ่งของฟังก์ชันคลื่นตรงบริเวณรอยต่อ (x = –a และ x = a) ต้องมีค่าต่อเนื่อง

ที่ x = –a ใช้เงื่อนไข $\psi_1(-a) = \psi_2(-a)$ จะได้

$$Ae^{-\ell a} = Ce^{-i\kappa a} + De^{i\kappa a} \qquad (2.10)$$

ใช้เงื่อนไข $\left.\frac{\partial \psi_1}{\partial x}\right|_{x=-a} = \left.\frac{\partial \psi_2}{\partial x}\right|_{x=-a}$ จะได้

$$A\ell e^{-\ell a} = i\kappa(Ce^{-i\kappa a} - De^{i\kappa a}) \qquad (2.11)$$

ที่ x = +a ใช้เงื่อนไข $\psi_2(a) = \psi_3(a)$ จะได้

$$Ge^{-\ell a} = Ce^{i\kappa a} + De^{-i\kappa a} \qquad (2.12)$$

ใช้เงื่อนไข $\left.\frac{\partial \psi_2}{\partial x}\right|_{x=a} = \left.\frac{\partial \psi_3}{\partial x}\right|_{x=a}$ จะได้

$$-G\ell e^{-\ell a} = i\kappa(Ce^{+i\kappa a} - De^{-i\kappa a}) \tag{2.13}$$

หาค่าของ C และ D ในเทอมของ A จะได้

$i\kappa(2.10)+(2.11)$ $$C = e^{(-\ell+i\kappa)a}(\frac{\ell+i\kappa}{2i\kappa})A \tag{2.14}$$

$(2.11)-i\kappa(2.10)$ $$-D = e^{-(\ell+i\kappa)a}(\frac{\ell-i\kappa}{2i\kappa})A \tag{2.15}$$

หาค่าของ C และ D ในเทอมของ G จะได้

$i\kappa(2.12)+(2.13)$ $$-C = e^{-(\ell+i\kappa)a}(\frac{\ell-i\kappa}{2i\kappa})G \tag{2.16}$$

$(2.13)-i\kappa(2.12)$ $$D = e^{(-\ell+i\kappa)a}(\frac{\ell+i\kappa}{2i\kappa})G \tag{2.17}$$

หาความสัมพันธ์ระหว่าง A และ G

จากสมการ (2.14) และ (2.16) ต่างเท่ากับ C จะได้

$$e^{(-\ell+i\kappa)a}(\frac{\ell+i\kappa}{2i\kappa})A \quad = \quad -e^{-(\ell+i\kappa)a}(\frac{\ell-i\kappa}{2i\kappa})G \tag{2.18}$$

จากสมการ (2.15) และ (2.17) ต่างเท่ากับ D จะได้

$$-e^{-(\ell+i\kappa)a}(\frac{\ell-i\kappa}{2i\kappa})A \quad = \quad e^{(-\ell+i\kappa)a}(\frac{\ell+i\kappa}{2i\kappa})G \tag{2.19}$$

$(2.18)\div(2.19)$

$$\frac{e^{(-\ell+ik)a}(\ell+i\kappa)}{e^{-(\ell+i\kappa)a}(\ell-i\kappa)} = \frac{e^{-(\ell+i\kappa)a}(\ell-i\kappa)}{e^{(-\ell+ik)a}(\ell+i\kappa)}$$

$$e^{i(2ka)} = \pm\frac{(\ell-i\kappa)}{(\ell+i\kappa)} \tag{2.20}$$

สมการ (2.20) แยกได้เป็น 2 กรณี คือ

$$\frac{\ell-i\kappa}{\ell+i\kappa} = -e^{i(2\kappa a)} \tag{2.21}$$

และ $$\frac{\ell - i\kappa}{\ell + i\kappa} = +e^{i(2\kappa a)} \qquad (2.22)$$

พิจารณากรณี $\frac{\ell - i\kappa}{\ell + i\kappa} = -e^{i(2\kappa a)}$

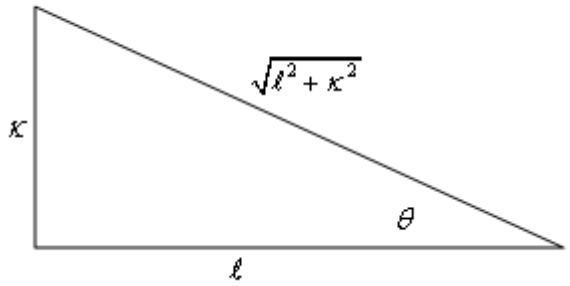
กำหนดให้ด้านซ้ายมือ $\frac{\ell - i\kappa}{\ell + i\kappa} = +e^{-i(2\theta)}$ เมื่อ θ คือมุมที่กำหนดขึ้นมาและจะนำไปหาความสัมพันธ์ระหว่าง θ กับ κa

หรือ $$\frac{\ell^2 - \kappa^2}{\ell^2 + \kappa^2} - i\frac{2\ell\kappa}{\ell^2 + \kappa^2} = \cos 2\theta - i\sin 2\theta \qquad (2.23)$$

จากสมการ (2.23) จะได้ $\cos 2\theta = \frac{\ell^2 - \kappa^2}{\ell^2 + \kappa^2}$

และจะได้ $\cos\theta = \sqrt{\frac{1}{2}(\cos 2\theta + 1)} = \pm\frac{\ell}{\sqrt{\ell^2 + \kappa^2}}$

นำไปสู่ $\sin\theta = \pm\frac{\kappa}{\sqrt{\ell^2 + \kappa^2}}$ และ $\cot\theta = \frac{\ell}{\kappa}$



ด้านขวามือของสมการ (2.21) $-e^{i(2ka)} = e^{-i2\theta}$

นั่นคือ $2\kappa a = \pi - 2\theta$

$$\theta = \frac{\pi}{2} - \kappa a$$

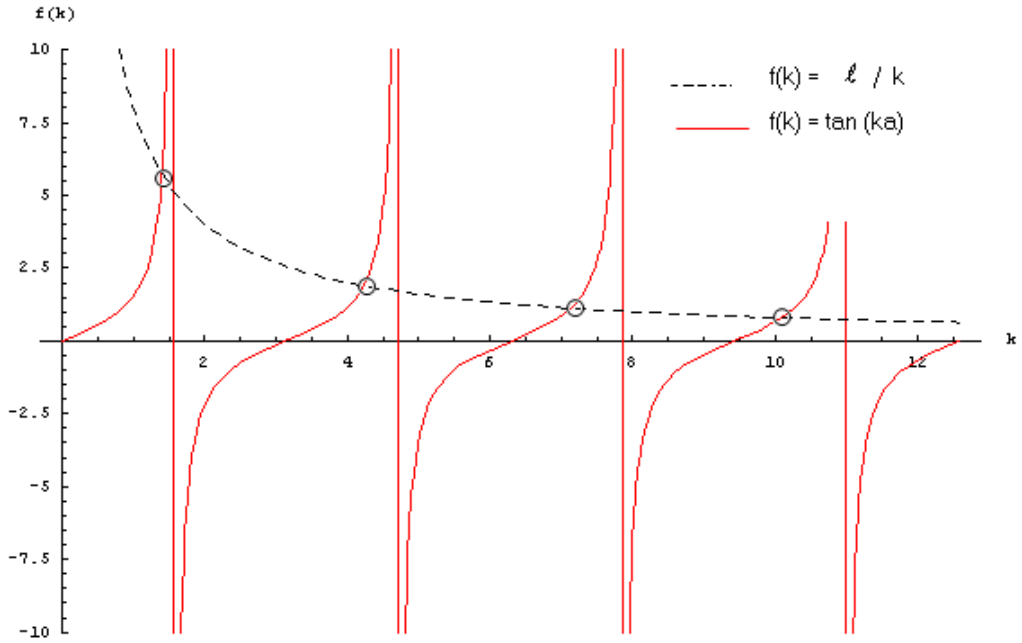
จาก $\cot\theta = \frac{\ell}{\kappa}$ แทนค่า θ

$$\cot\theta = \cot(\frac{\pi}{2} - \kappa a) = \tan(\kappa a) = \frac{\ell}{\kappa}$$

$$\tan(\kappa a) = \frac{\ell}{\kappa} \qquad (2.24)$$

เป็นจริงทุก ๆ ค่าของ κ โดยที่ $\tan(\kappa a)$ มีค่ามากกว่าหรือเท่ากับ 0

สมการที่ (2.24) สามารถนำไปใช้หาพลังงานของอนุภาคที่เป็นไปได้ภายในบ่อศักย์ เพราะ κ เป็นฟังก์ชันกับ E แต่การหาค่า E ไม่สามารถทำได้โดยง่าย เพราะสมการที่ (2.24) เป็นฟังก์ชันอดิศัย (transcendental function) ตัวแปร κ ปรากฏอยู่ในค่า tan ด้านซ้ายมือ และยังปรากฏให้เห็นด้านขวามืออีกด้วย การหารากสมการนี้ต้องทำโดยวิธีคำนวณเชิงตัวเลข (numerical method) หรือ ใช้วิธีพล็อตกราฟ ระหว่าง $\tan \kappa a$ กับ $\frac{\ell}{\kappa}$ บนแกนเดียวกัน เพื่อหาจุดตัดของกราฟ



รูปที่ 2.2 กราฟแสดงความสัมพันธ์ระหว่าง tan (κ a) และ $\frac{\ell}{\kappa}$ กับ κ

จุดที่ตัดกันนั้นคือจุดที่ κ มีค่าเป็นไปได้สอดคล้องกับสมการที่ (2.24) เมื่อรู้ค่า κ สามารถนำไปหาพลังงานของอนุภาคได้ ซึ่งแสดงให้เห็นว่า พลังงานสามารถมีค่าได้เพียงบางค่าหรือมีค่าไม่ต่อเนื่อง

ทดลองตรวจสอบผลลัพธ์ที่ได้โดย พิจารณากรณีที่ $V \to \infty$ ค่า $\tan \kappa a \to 0$ นั่นคือ $\kappa a = n\,\pi$ ฟังก์ชันคลื่น ψ จะหายไปที่ผนังบ่อ พลังงานของอนุภาค E หาได้จาก

$$E = \frac{\bar{h}^2 \kappa^2}{2m} = \frac{\bar{h}^2 (\frac{n\pi}{a})^2}{2m} = \frac{\bar{h}^2 n^2 \pi^2}{2ma^2}$$

ผลลัพธ์นี้สอดคล้องกับระดับพลังงานของอนุภาคที่ถูกกักอยู่ในบ่อศักย์สี่เหลี่ยมที่มีความสูงอนันต์

เพื่อให้ง่ายต่อการเขียนกราฟและอภิปรายผล จึงเปลี่ยนรูปของสมการ (2.24) ให้อยู่ในรูปของ cosine ดังนี้

$$\tan^2 \kappa a + 1 = \frac{\ell^2 + \kappa^2}{\kappa^2} = \sec^2 \kappa a = \frac{1}{\cos^2 ka} \qquad (2.25)$$

เมื่อแทนค่า ℓ และ κ จากสมการ (2.6) และ (2.7) ลงในเทอม $\ell^2 + \kappa^2$

$$\text{จะได้} \quad \ell^2 + \kappa^2 = \frac{2mV(x)}{\hbar^2} = \kappa_0^2 \qquad (2.26)$$

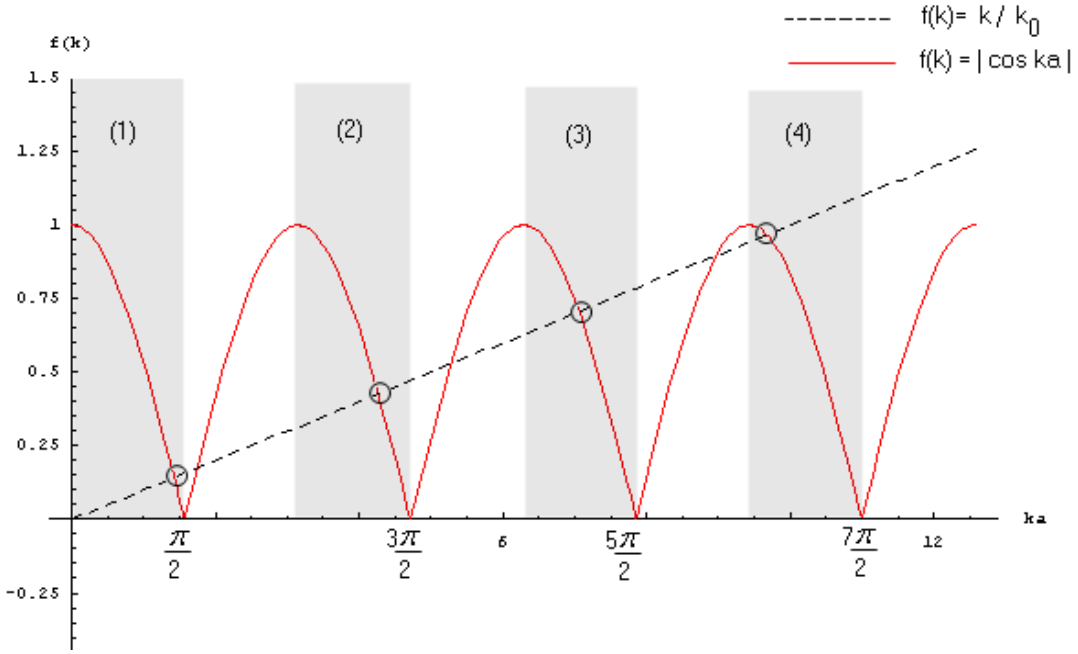
κ_0 จะเป็นตัวบอกความลึกของบ่อศักย์

สมการ (2.25) จะได้ผลลัพธ์เป็น

$$|\cos \kappa a| = \frac{\kappa}{\kappa_0} \qquad (2.27)$$

สมการ (2.27) เป็นจริงสำหรับทุกค่าของ κ โดยที่ $\tan(\kappa a) \geq 0$

เขียนกราฟระหว่าง $|\cos \kappa a|$ กับ $\frac{\kappa}{\kappa_0}$ บนแกนเดียวกัน เพื่อหาจุดตัดของกราฟ



รูปที่ 2.3 เปลี่ยนรูปฟังก์ชัน tan ให้อยู่ในรูป cos เพื่อจะได้ดูง่ายขึ้น

บริเวณที่แรเงา (1) (2) (3) และ (4) เป็นบริเวณที่ $\tan \kappa a \geq 0$ จากค่า κ_0 ที่กำหนดให้ จะได้ค่า κ ที่เป็นรากสมการของสมการ (2.27) จำนวน 4 ค่า ถ้า κ_0 เพิ่มค่ามากขึ้น จะได้จำนวนรากสมการเพิ่มขึ้นตามไปด้วย จะเห็นว่าทุก ๆ ค่า κ_0 จะต้องมีรากสมการหรือมีระดับพลังงานของ Bound state อย่างน้อย 1 ค่าเสมอ

พิจารณากรณี $\frac{\ell - i\kappa}{\ell + i\kappa} = e^{-i(2\kappa a)}$

กำหนดให้ด้านซ้ายมือ $\frac{\ell - i\kappa}{\ell + i\kappa} = +e^{-i(2\theta)}$ เช่นเดียวกันกับกรณีแรก

นั่นคือ

$$e^{-i(2\theta)} = e^{i(2\kappa a)}$$

$$2\theta = -2\kappa a$$

เพราะ $\cot\theta = \frac{\ell}{\kappa}$ แทนค่า θ จะได้

$$\cot(-\kappa a) = -\cot(\kappa a) = \frac{\ell}{\kappa}$$

$$-\cot(\kappa a) = \frac{\ell}{\kappa} \qquad (2.28)$$

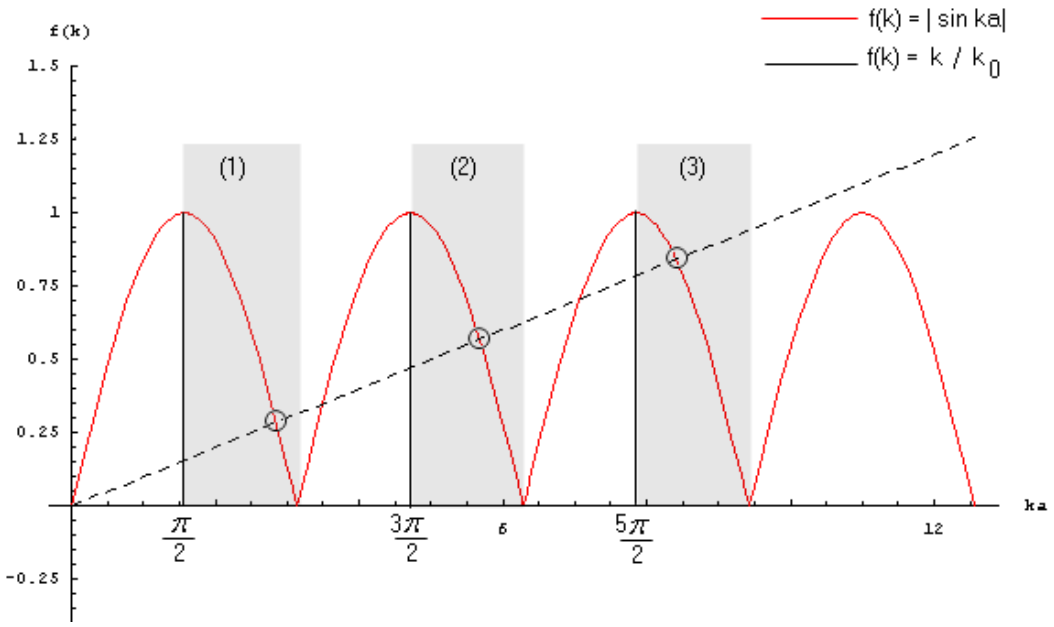
สมการ (2.28) เป็นจริงสำหรับทุก ๆ ค่าของ κ โดยที่ $\cot(\kappa a) \le 0$

เพื่อให้ง่ายต่อการพล็อตกราฟ จึงเป็นรูปสมการ (2.28) ให้เป็นฟังก์ชันของ sine ดังนี้

$$1 + \cot^2 \kappa a = \frac{\kappa^2 + \ell^2}{\kappa^2} = \frac{\kappa_0^2}{\kappa^2} = \cos ec^2 \kappa a = \frac{1}{\sin^2 \kappa a}$$

$$\left|\sin \kappa a\right| = \frac{\kappa}{\kappa_0} \qquad (2.29)$$

เขียนกราฟระหว่าง $|\sin \kappa a|$ กับ $\frac{\kappa}{\kappa_0}$ บนแกนเดียวกัน เพื่อหาจุดตัดของกราฟ โดยเลือกจุดตัดที่เกิดขึ้นตรงบริเวณที่ $\cot(\kappa a) \le 0$



รูปที่ 2.4 เปลี่ยนรูปฟังก์ชัน cot ให้อยู่ในรูป sin เพื่อจะได้ดูง่ายขึ้น

นำค่า κ ที่หาได้ไปแทนค่าลงในสมการ $e^{i(2ka)} = \pm\frac{(\ell - i\kappa)}{(\ell + i\kappa)}$ (สมการ (2.20)) เพื่อหาความสัมพันธ์ระหว่าง A, C, D และ G ได้ดังนี้

เมื่อ $\frac{\ell - i\kappa}{\ell + i\kappa} = -e^{i(2\kappa a)}$ จะได้ C = D และ A = G ทำให้ได้ฟังก์ชันคลื่นที่มีคุณสมบัติ $\psi(-x) = \psi(x)$ ผลเฉลยที่ได้จะเป็น ฟังก์ชันคู่ (Even function)

เมื่อ $\frac{\ell - i\kappa}{\ell + i\kappa} = +e^{i(2\kappa a)}$ จะได้ C = –D และ A = –G ทำให้ได้ฟังก์ชันคลื่นที่มีคุณสมบัติ $\psi(-x) = -\psi(x)$ ผลเฉลยที่ได้จะเป็น ฟังก์ชันคี่ (Odd function)

จากสมการ (2.24) และ (2.28) ทำให้เกิดฟังก์ชันคลื่นที่เป็นฟังก์ชันคู่และคี่ตามลำดับ นำความรู้ทางตรีโกณมิติ รวมสมการทั้งสองเป็นสมการเดียวกัน ดังนี้

จากสมการ (2.28) ซึ่งมีรูปสมการเป็น $-\cot(\kappa a) = \frac{\ell}{\kappa}$ เขียนให้อยู่ในรูป tan โดยอาศัยความสัมพันธ์ $\tan(\theta - \frac{\pi}{2}) = -\cot\theta$ สมการ (2.28) จะกลายเป็น

$$\tan(\kappa a - \frac{\pi}{2}) = \frac{\ell}{\kappa} \qquad (2.30)$$

เมื่อนำสมการ (2.24) ไปพล็อตกราฟ จะได้ค่า tan เป็นช่วง ๆ ดังรูปที่ 2.2 โดยอาศัยความสัมพันธ์ทางตรีโกณมิติ

$$\tan(\theta - \pi) = \tan(\theta - 2\pi) = \tan(\theta - 3\pi)... = \tan(\theta - m\pi) = \tan(\theta)$$

เมื่อ m = 0,1,2,3...

จึงเขียนสมการ (2.24) ให้อยู่ในรูปทั่วไปได้เป็น

$$\tan(\kappa a - m\pi) = \frac{\ell}{\kappa}$$

หรือ
$$\kappa a = m\pi + \tan^{-1}\frac{\ell}{\kappa} \qquad (2.31)$$

และเขียนสมการ (2.30) อยู่ในรูปทั่วไปได้เช่นกัน

$$\tan(\kappa a - \frac{\pi}{2} - m\pi) = \frac{\ell}{\kappa}$$

หรือ
$$\kappa a = \frac{\pi}{2} + m\pi + \tan^{-1}\frac{\ell}{\kappa} \qquad (2.32)$$

สมการ (2.31) คือสมการที่ทำให้ผลเฉลยเป็นฟังก์ชันคู่และ สมการ (2.32) คือสมการที่ทำให้ผลเฉลยเป็นฟังก์ชันคี่ เมื่อ m = 0, 1, 2, 3, ... รูปสมการทั้งสองมีลักษณะคล้ายกัน ต่างกัน

ตรงค่า $\frac{\pi}{2}$ ต้องการรวมสมการทั้งสองให้เขียนเป็นสมการเดียวกันเพื่อความกระทัดรัด โดยจัดเงื่อนไขให้เหมาะสม จนสามารถสอดคล้องกับความเป็นฟังก์ชันคู่และคี่ของสมการได้

ถ้ากำหนดให้ m = $\frac{1}{2}(n-1)$ สามารถเขียนรวมสมการ (2.31) และ (2.32) ให้เหลือสมการเดียวดังนี้

$$\kappa a = \frac{1}{2}(n-1)\pi + \tan^{-1}\frac{\ell}{\kappa}$$

$$2\kappa a = (n-1)\pi + 2\tan^{-1}\frac{\ell}{\kappa} \qquad (2.33)$$

เมื่อ n = 1,2,3,... เป็นจำนวนเต็มบวกที่เริ่มต้นด้วย 1

เมื่อ n = 1,3,5,7,9....จะสอดคล้องกับสมการ (2.31) หรือผลเฉลยที่ได้จะเป็นฟังก์ชันคู่

เมื่อ n = 2,4,6,8,10 ... จะสอดคล้องกับสมการ (2.32) ผลเฉลยที่ได้จะเป็นฟังก์ชันคี่

ค่า n นี้คือเลขควอนตัม ที่แสดงสถานะของพลังงานของอนุภาคในระบบ จะเห็นว่า n ที่เป็นเลขคี่ จะให้ฟังก์ชันคลื่นที่เป็นฟังก์ชันคู่ ในทางกลับกัน n ที่เป็นเลขคู่ จะให้ฟังก์ชันคลื่นที่เป็นฟังก์ชันคี่ สมการ (2.33) ทำให้มองภาพการหาพลังงาน (E) ได้ชัดเจนขึ้น เมื่อแทนค่า κ และ ℓ ลงไป จะได้เป็น

$$2a\sqrt{\frac{2m}{\hbar^2}(V_0 - E)} = (n-1)\pi + 2\tan^{-1}\sqrt{\frac{E}{V_0 - E}} \qquad (2.34)$$

ในหน่วยอะตอม สมการจะถูกลดรูปเหลือเพียง

$$2a\sqrt{2(V_0 - E)} = (n-1)\pi + 2\tan^{-1}\sqrt{\frac{E}{V_0 - E}}$$

สมการที่ (2.34) เป็นฟังก์ชันอดิศัย (transcendental function) จะเห็นว่าตัวแปร E ไม่สามารถเขียนแสดงความสัมพันธ์กันได้อย่างชัดเจน มี E ปรากฏอยู่ทางด้านซ้ายมือ และยังมี E ปรากฏให้เห็นด้านขวามืออีกด้วย การหาค่า E ไม่สามารถทำได้โดยวิธีวิเคราะห์ ต้องใช้วิธีการคำนวณเชิงตัวเลขเข้ามาช่วย

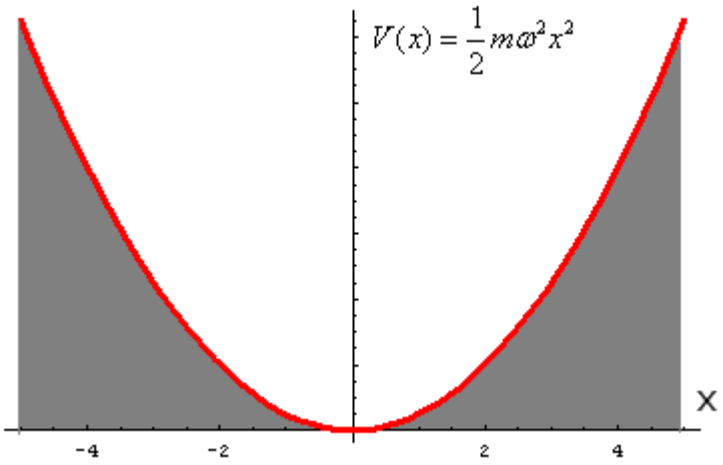
2.2.2 ฮาร์มอนิก ออสซิลเลเตอร์ แบบ 1 มิติ (Harmonic Oscillator)

สมการชเรอดิงเงอร์เขียนได้เป็น

$$-\frac{\bar{h}^2}{2m}\frac{\partial^2\psi(x)}{\partial x^2}+\frac{1}{2}m\omega^2x^2\psi(x)=E\psi(x) \qquad (2.35)$$

เมื่อ $V(x)=\frac{1}{2}m\omega^2x^2$

ω คือความเร็วเชิงมุมของการสั่นของอนุภาคในหน่วย radian/sec



รูปที่ 2.5 ฟังก์ชันของ Harmonic oscillator

กำหนดให้ $\varepsilon=\frac{2E}{\bar{h}\omega}$ และ $\xi=\sqrt{\frac{m\omega}{\bar{h}}}x$

โดยใช้กฎลูกโซ่ (chain rule)

$$\frac{d\psi}{dx}=\frac{d\psi}{d\xi}\frac{d\xi}{dx}=\sqrt{\frac{m\omega}{\bar{h}}}\frac{d\xi}{dx}$$

$$\frac{d^2\psi}{dx^2}=\frac{d}{dx}\frac{d\psi}{dx}=\sqrt{\frac{m\omega}{\bar{h}}}\frac{d}{d\xi}\left(\frac{d\psi}{d\xi}\frac{d\xi}{dx}\right)=\sqrt{\frac{m\omega}{\bar{h}}}\frac{d^2\psi}{d\xi^2}$$

แทนค่าลงในสมการ (2.35) จะเหลือสั้น ๆ เพียง

$$\frac{d^2\psi_n}{d\xi^2}+\left(\varepsilon-\xi^2\right)\psi_n=0 \qquad (2.36)$$

ในหนังสือกลศาสตร์ควอนตัมเกือบทุกเล่ม จะหาคำตอบของสมการอนุพันธ์ สมการ (2.36) นี้โดยการหาค่า ψ ที่ ξ มีค่ามาก ๆ สู่อนันต์ ที่เรียกว่า asymptotic จากนั้นจึงกลับมาหาค่า ψ ที่ ξ มีค่าน้อย ๆ ซึ่งจะได้ผลเฉลยทั่วไปของสมการอนุพันธ์นี้คือ

$$\psi(\xi) = H(\xi)e^{-\frac{\xi^2}{2}}$$

ในที่นี้จะใช้ตัวปฏิบัติการดิฟเฟอเรนเชียล (Differential operator) หาค่าพลังงานของระบบ จัดรูปสมการ (2.36) ใหม่ โดยแทนค่า $\varepsilon = \frac{2E}{\hbar\omega}$ กลับคืน จะได้

$$\left(-\frac{d^2}{d\xi^2} + \xi^2\right)\psi_n = \frac{2E_n}{\hbar\omega}\psi_n \qquad (2.37)$$

จากคุณสมบัติของ Differential operator

$$\left(\xi - \frac{d}{d\xi}\right)\left(\xi + \frac{d}{d\xi}\right) = \xi^2 - \frac{d^2}{d\xi^2} - \frac{d}{d\xi}\xi + \xi\frac{d}{d\xi}$$

$$= \xi^2 - \frac{d^2}{d\xi^2} - 1 - \xi\frac{d}{d\xi} + \xi\frac{d}{d\xi}$$

$$= \xi^2 - \frac{d^2}{d\xi^2} - 1 \qquad (2.38)$$

และ

$$\left(\xi + \frac{d}{d\xi}\right)\left(\xi - \frac{d}{d\xi}\right) = \xi^2 - \frac{d^2}{d\xi^2} + \frac{d}{d\xi}\xi - \xi\frac{d}{d\xi}$$

$$= \xi^2 - \frac{d^2}{d\xi^2} + 1 + \xi\frac{d}{d\xi} - \xi\frac{d}{d\xi}$$

$$= \xi^2 - \frac{d^2}{d\xi^2} + 1 \qquad (2.39)$$

ใช้ความสัมพันธ์ในสมการ (2.38) และ (2.39) สามารถเปลี่ยนรูปสมการ (2.37) ได้ 2 แบบคือ

แบบที่ 1

$$\left[\left(\xi - \frac{d}{d\xi}\right)\left(\xi + \frac{d}{d\xi}\right) + 1\right]\psi_n = \frac{2E}{\hbar\omega}\psi_n$$

หรือ $$\left[\left(\xi-\frac{d}{d\xi}\right)\left(\xi+\frac{d}{d\xi}\right)\right]\psi_n=\left(\frac{2E}{\bar{h}\omega}-1\right)\psi_n \qquad (2.40)$$

แบบที่ 2

$$\left[\left(\xi+\frac{d}{d\xi}\right)\left(\xi-\frac{d}{d\xi}\right)-1\right]\psi_n=\frac{2E}{\bar{h}\omega}\psi_n$$

หรือ $$\left[\left(\xi+\frac{d}{d\xi}\right)\left(\xi-\frac{d}{d\xi}\right)\right]\psi_n=\left(\frac{2E}{\bar{h}\omega}+1\right)\psi_n \qquad (2.41)$$

จากสมการ (2.40)

ให้ $$\phi_n=\left(\xi+\frac{d}{d\xi}\right)\psi_n \qquad (2.42)$$

สมการที่ (2.40) จะเหลือเพียง

$$\left(\xi-\frac{d}{d\xi}\right)\phi_n=\left(\frac{2E}{\bar{h}\omega}-1\right)\psi_n$$

เมื่อใช้ Hamiltonian operator กระทำกับ ϕ_n เพื่อหาค่าไอเกนของระบบ

$$\begin{aligned}\hat{H}\phi_n&=\left[\left(\xi+\frac{d}{d\xi}\right)\left(\xi-\frac{d}{d\xi}\right)-1\right]\phi_n\\&=\left(\xi+\frac{d}{d\xi}\right)\left[\left(\xi-\frac{d}{d\xi}\right)\left(\xi+\frac{d}{d\xi}\right)-1\right]\psi_n\\&=\left(\xi+\frac{d}{d\xi}\right)\left[\left(\frac{2E}{\bar{h}\omega}-1\right)-1\right]\psi_n\\&=\left(\frac{2E_n}{\bar{h}\omega}-2\right)\phi_n\end{aligned}$$

จะเห็นว่าค่าไอเกนที่สอดคล้องกับสมการชเรอดิงเงอร์คือ $\left(\frac{2E_n}{\bar{h}\omega}-2\right)$

ในทำนองเดียวกันจากสมการ (2.41)

ให้ $$\theta_n=\left(\xi-\frac{d}{d\xi}\right)\psi_n \qquad (2.43)$$

เมื่อใช้ Hamiltonian operator กระทำกับ θ_n เพื่อหาค่าไอเกนของระบบ จะได้

$$\hat{H}\theta_n=\left(\frac{2E_n}{\bar{h}\omega}+2\right)\theta_n$$

จะเห็นว่า $\left(\xi+\frac{d}{d\xi}\right)$ เป็น operator ที่ลดค่า (lower) พลังงานของระบบ และ $\left(\xi-\frac{d}{d\xi}\right)$ เป็น operator ที่เพิ่มค่า (raise) พลังงานของระบบ

แต่การลดค่าพลังงาน สามารถลดค่าได้ไม่น้อยกว่าพลังงานต่ำสุดของบ่อศักย์ สำหรับฮาร์มอนิก ออสซิลเลเตอร์ มีค่าพลังงานต่ำสุดเท่ากับ ศูนย์ ดังนั้น

$$\left(\xi+\frac{d}{d\xi}\right)\psi_0=0$$

$$\frac{d\psi_0}{d\xi}=-\xi\psi_0$$

หาปริพันธ์ทั้งสองข้าง เพื่อหาฟังก์ชันคลื่น ψ_0

$$\int\frac{d\psi_0}{\psi_0}=-\int\xi d\xi$$

$$\ln\psi_0=-\frac{\xi^2}{2}+\ln A$$

$$\psi_0=Ae^{-\frac{\xi^2}{2}}=Ae^{-\frac{m\omega x^2}{2\bar{h}}}$$

เมื่อ A คือค่าคงที่ใดๆ ฟังก์ชันคลื่นนี้เป็นฟังก์ชันคลื่นที่ยังไม่ได้ทำการ normalize ต้องการหาค่า E_0 ทำได้โดยแทนค่า ψ_0 ลงในสมการ (2.38)

$$\left[\left(\xi-\frac{d}{d\xi}\right)\left(\xi+\frac{d}{d\xi}\right)\right]\psi_0=\left(\frac{2E}{\bar{h}\omega}-1\right)\psi_0$$

ด้านซ้ายมือมีค่าเป็นศูนย์ เพราะ $\left(\xi+\frac{d}{d\xi}\right)\psi_0=0$

จึงได้ $$\left(\frac{2E}{\bar{h}\omega}-1\right)=0$$

$$E_0=\frac{\bar{h}\omega}{2}$$

เริ่มต้นจาก ψ_0 สามารถใช้ raising operator โดยกำหนดให้ $\frac{1}{\sqrt{2}}\left(\xi-\frac{d}{d\xi}\right)=a_+$ หาค่าพลังงานที่ระดับพลังงาน n ต่าง ๆ กันได้

จากสมการ (2.43) $$\theta_n=\left(\xi-\frac{d}{d\xi}\right)\psi_n$$

$$\theta_0 = a_+\psi_0 = \frac{1}{\sqrt{2}}\left(\xi - \frac{d}{d\xi}\right)\psi_0 = C_0\psi_1$$

และใช้ lower operator โดยกำหนดให้ $\frac{1}{\sqrt{2}}\left(\xi + \frac{d}{d\xi}\right) = a_-$ หาพลังงานที่เลขควอนตัม n มีค่าต่ำลง

จากสมการ (2.42) $\phi_n = \left(\xi + \frac{d}{d\xi}\right)\psi_n$

$$\phi_1 = a_-\psi_1 = \frac{1}{\sqrt{2}}\left(\xi + \frac{d}{d\xi}\right)\psi_1 = D_1\psi_0$$

เมื่อ C_0 และ D_0 เป็นค่าคงที่ การใช้ operator กระทำกับฟังก์ชันคลื่นเช่นนี้ ทำให้ได้เซตของพลังงานที่เลขควอนตัม n ใด ๆ ชุดหนึ่งดังนี้

$$E_n = \bar{h}\omega(n + \frac{1}{2}) \qquad (2.44)$$

เมื่อ n = 0, 1, 2, 3, ...

ในการเขียนโปรแกรมเพื่อจำลองอนุภาคในบ่อศักย์ ต้องการให้ n เริ่มต้นที่ 1, 2, 3, ... จึงแทนค่า n ด้วย n–1 สมการ (2.44) จะกลายเป็น

$$E_{n-1} = \bar{h}\omega(n - \frac{1}{2}) \qquad (2.45)$$

เมื่อ n = 1, 2, 3, ...

ขั้นตอนการหาพลังงานโดยใช้ตัวปฏิบัติการทั้งหมด สามารถนำไปเปรียบเทียบกับการใช้ตัวปฏิบัติการที่พบในหนังสือกลศาสตร์ควอนตัมทั่วไป ดังนี้

ให้ $p = -i\bar{h}\frac{d}{dx}$

เขียนสมการชเรอดิงเงอร์ให้อยู่ในรูปสมการพลังงานได้ดังนี้

$$\frac{1}{2m}\left(p^2 + (m\omega x)^2\right)\psi_n = E_n\psi_n$$

p และ x ไม่เป็นไปตามกฎการสลับที่ นั่นคือ

$$[x, p] = xp - px = \frac{\bar{h}}{i}\left(x\frac{d}{dx} - \frac{d}{dx}x\right)$$

$$= \frac{\bar{h}}{i}\left(x\frac{d}{dx} - x\frac{d}{dx} - 1\right) = i\bar{h}$$

$$\frac{1}{2m}\left(p^2 + (m\omega x)^2\right)\psi_n = \frac{1}{2m}\left((m\omega x + ip)(m\omega x - ip) - m\bar{h}\omega\right)\psi_n$$

$$= \frac{1}{2m}\left((m\omega x - ip)(m\omega x + ip) + m\bar{h}\omega\right)\psi_n$$

นิยามให้ $a_{\pm} = \frac{1}{\sqrt{2m\bar{h}\omega}}\left(\mp ip + m\omega x\right)$ จะสอดคล้องกับ $a_{\pm} = \frac{1}{\sqrt{2}}\left(\xi \mp \frac{d}{d\xi}\right)$ ที่ได้แสดงไว้ในข้างต้น

2.2.3 บ่อศักย์แบบอื่น ๆ

บ่อศักย์แบบสามเหลี่ยมสมมาตร (Symmetric triangular well) เป็นบ่อศักย์ที่เป็นรูปสามเหลี่ยมที่สมมาตรในแนวแกน y ผนังของบ่อศักย์จะมีลักษณะคล้ายกับผนังของบ่อศักย์ที่เกิดจากฮาร์มอนิก ออสซิลเลเตอร์ ก้นหลุมจะอยู่ที่จุดกำเนิด และพลังงานศักย์จะมีค่าเป็นบวก ต่างกันคือผนังของบ่อศักย์แบบสามเหลี่ยมจะลาดชันเป็นเส้นตรง สมการของบ่อศักย์ มีดังนี้

$$V(x) = V_0 |x|$$

บ่อศักย์แบบส่วนกลับของกำลังสองของไฮเปอร์บอลิก โคไซน์ (Inverse square hyperbolic cosine well) รูปของบ่อจะมีลักษณะคล้ายกับแบบฮาร์มอนิก ออสซิลเลเตอร์ (1 มิติ) แต่มีค่าเป็นลบ โดยที่ความลึกของบ่อคือ V_0 สมการของบ่อศักย์มีลักษณะดังนี้

$$V(x) = -\frac{V_0}{\cosh^2(\alpha x)}$$

บ่อศักย์แบบแทนเจนต์กำลังสอง (Square tangent Well) เป็นบ่อศักย์ที่เกิดจากค่ากำลังสองของแทนเจนต์ จะมีลักษณะคล้ายกับบ่อศักย์รูปสี่เหลี่ยม. แต่ก้นหลุมจะมีพลังงานศักย์เป็นศูนย์ สมการของบ่อศักย์ คือ

$$V(x) = V_0 \tan^2 ax$$

โดยที่ $-\frac{\pi}{2} \le x \le \frac{\pi}{2}$

2.3 วิธีการคำนวณเชิงตัวเลขที่ใช้ในงานวิจัย

ในการหาผลเฉลยของสมการชเรอดิงเงอร์แบบ 1 มิติ ไม่ขึ้นกับเวลาตามสมการ (2.3) ที่มีรูปสมการ ดังนี้

$$\frac{\partial^2\psi(x)}{\partial x^2} + 2\left(E - V(x)\right)\psi(x) = 0$$

ค่า V(x) จะเปลี่ยนไปตามบ่อศักย์ที่เลือกใช้ สมการนี้เป็นสมการอนุพันธ์อันดับสอง การหาผลเฉลยซึ่งก็คือฟังก์ชันคลื่นของอนุภาค $\psi(x)$ แล้วนำไปสู่การหาค่าพลังงาน E ซึ่งเป็นค่าเจาะจง (Eigen value) และระดับพลังงาน n ซึ่งมีค่าไม่ต่อเนื่อง วิธีการคำนวณเชิงตัวเลขที่ต้องนำมาใช้มีดังนี้

2.3.1 การหาผลเฉลยของสมการอนุพันธ์อันดับสอง โดยวิธีนูเมอรอฟ (Numerov's method)

Boris Vasil'evich Numerov. เป็นผู้คิดวิธีการหาผลเฉลยของสมการอนุพันธ์ ในปี 1927 โดยที่สมการอนุพันธ์จะต้องมีอันดับสองอยู่ในรูปแบบดังนี้

$$\frac{d^2y}{dx^2} + f(x)y = g(x) \qquad (2.46)$$

ให้ f(x) และ g(x) เป็นฟังก์ชันที่มีค่าต่อเนื่องในช่วง [a, b] แบ่งค่า x ในช่วง a ถึง b นี้ออกเป็นส่วนย่อยเท่า ๆ กัน แต่ละส่วนมีค่าเท่ากับ h

จัดรูปสมการ (2.46) ใหม่จะได้เป็น

$$\frac{d^2y}{dx^2} = -(f(x)y(x) - g(x))$$

เขียนสั้น ๆ ได้เป็น $y_i'' = -(f_i y_i - g_i)$ (2.47)

เมื่อ $f_i = f(x_i)$ $y_i = y(x_i)$ และ $g_i = g(x_i)$

ให้ $\Delta x = h = x_i - x_{i-1}$

เริ่มต้นด้วยการกระจาย $y(x_i)$ เป็นอนุกรมเทเลอร์ (Taylor series) รอบ ๆ จุด x_i

$$y_{i+1} = y(x_i + h) = y(x_i) + hy'(x_i) + \frac{h^2}{2!}y''(x_i) + \frac{h^3}{3!}y'''(x_i) + \frac{h^4}{4!}y''''(x_i) + \frac{h^5}{5!}y'''''(x_i) + \ldots$$

$$y_{i-1} = y(x_i - h) = y(x_i) - hy'(x_i) + \frac{h^2}{2!}y''(x_i) - \frac{h^3}{3!}y'''(x_i) + \frac{h^4}{4!}y''''(x_i) - \frac{h^5}{5!}y'''''(x_i) + \ldots$$

นำสมการทั้งสองบวกเข้าด้วยกัน

$$y_{i+1} + y_{i-1} = 2y_i + h^2 y_i'' + \frac{h^4}{12}y_i'''' + O(h^6)$$

แทนค่า $y_i'' = -(f_i y_i - g_i)$ จากสมการ (2.47)

$$h^2(f_i y_i - g_i) = 2y_n - y_{n+1} - y_{n-1} + \frac{h^4}{12}y_i'''' + O(h^6) \qquad (2.48)$$

จากสมการ (2.47) หาอนุพันธ์ 2 ครั้ง

$$y''''(x) = -\frac{d^2}{dx}(f(x)y(x) - g(x))$$

จากนิยามการหาอนุพันธ์อันดับสอง ที่ใช้ในวิชาแคลคูลัส

$$f''(x) = \frac{f(x+h) - 2f(x) + f(x-h)}{h^2}$$

ดังนั้น

$$y''''(x_i) = -\frac{d^2}{dx}(f(x_i)y(x_i) - g(x_i)) = -\frac{(f_{i+1}y_{i+1} - g_{i+1}) - 2(f_i y_i - g_i) + (f_{i-1}y_{i-1} - g_i)}{h^2} \quad (2.49)$$

แทนค่า y'''' จากสมการ (4) ลงในสมการ (2.48)

$$h^2(f_i y_i - g_i) = 2y_i - y_{i+1} - y_{i-1} - \frac{h^4}{12}(\frac{(f_{i+1}y_{i+1} - g_{i+1}) - 2(f_i y_i - g_i) + (f_{i-1}y_{i-1} - g_{i-1})}{h^2}) + O(h^6)$$

ต้องการหาค่า y_{i+1} จัดเทอมสมการบน เสียใหม่

$$\left(1 + \frac{h^2}{12} f_{i+1}\right) y_{i+1} = 2\left(1 - \frac{5h^2}{12} f_i\right) y_i - \left(1 + \frac{h^2}{12} f_{i-1}\right) y_{i-1} + \frac{h^2}{12}\left(g_{i+1} + 10g_i + g_{i-1}\right) + O(h^6) \quad (2.50)$$

สมการ (2.50) คือสมการที่ใช้ในการหาผลเฉลยของสมการอนุพันธ์อันดับสอง จากสมการชเรอดิงเงอร์ที่ไม่ขึ้นกับเวลา มีรูปสมการเป็น

$$\frac{\partial^2 \psi(x)}{\partial x^2} + \frac{2m}{\overline{h}^2}(E - V(x))\psi(x) = 0$$

ให้ $k^2(x) = \frac{2m}{\overline{h}^2}(E - V)$

จะได้ $$\frac{\partial^2 \psi(x)}{\partial x^2} + k^2(x)\psi(x) = 0$$

ซึ่งมีรูปสมการสอดคล้องกับสมการ (2.46) โดยที่ k^2 (x) เทียบได้กับ f(x) และ g(x) =0 จึงสามารถใช้วิธีนูเมอรอฟมาใช้แก้ปัญหาได้ โดยตัดเทอม g(x) ทิ้งไป สมการ (2.50) จะเหลือเพียง

$$(1 + \frac{h^2}{12} f_{i+1}) y_{i+1} = (2 - \frac{5h^2}{6} f_i) y_i - (1 + \frac{h^2}{12} f_{i-1}) y_{i-1} + O(h^6)$$

$$y_{i+1} = \frac{(2 - \frac{5h^2}{6} f_i) y_i - (1 + \frac{h^2}{12} f_{i-1}) y_{i-1}}{(1 + \frac{h^2}{12} f_{i+1})} + O(h^6)$$

จะเห็นได้ว่าอนุกรมจะลู่เข้า(converge) ที่เทอมอันดับที่ 4 จึงตัดเทอม h^6 ทิ้ง เมื่อนำตัวแปรในสมการชเรอดิงเงอร์ เทียบกับสมการ (2.50) จะได้

$$\psi_{i+1} = \frac{(2 - \frac{5h^2}{6} k_i^2) \psi_i - (1 + \frac{h^2}{12} k_{i-1}^2) \psi_{i-1}}{(1 + \frac{h^2}{12} k_{i+1}^2)} \qquad (2.51)$$

ในการคำนวณเชิงตัวเลข ให้ $x_0, x_1, x_2, ..x_i, x_{i+1}, x_{i+1}, ..., x_n$ หมายถึงจุดต่าง ๆ ซึ่งมีค่าอยู่ระหว่าง x =a ถึง x = b ต้องกำหนดเงื่อนไขขอบเขต (Boundary condition) มาให้ ต้องทราบค่า ψ_i, ψ_{i-1} จึงจะหาค่า ψ_{i+1} ได้

2.3.2. การหารากสมการโดยวิธีแบ่งครึ่งช่วง (Bisection method)

การหาค่าพลังงานของอนุภาค E ในบ่อศักย์ ดังสมการ (2.34) ไม่สามารถใช้วิธีวิเคราะห์หาได้โดยตรง การตรวจสอบผลลัพธ์ที่คำนวณได้จากการจำลองสถานการณ์ จึงต้องใช้วิธีการหารากสมการโดยวิธีแบ่งครึ่งช่วงหาผลลัพธ์เปรียบเทียบ จากสมการ (2.34)

$$2a\sqrt{2(V_0 - E)} = (n-1)\pi + 2\tan^{-1}\sqrt{\frac{E}{V_0 - E}}$$

เมื่อ 2a คือความกว้างของบ่อศักย์ V_0 คือความลึกของบ่อศักย์ซึ่งกำหนดมาให้ จัดรูปสมการใหม่ดังนี้

$$f(E) = 2a\sqrt{2(V_0 - E)} - (n-1)\pi - 2\tan^{-1}\sqrt{\frac{E}{V_0 - E}}$$

ถ้า E เป็นรากสมการ เมื่อแทนค่าลงไปในสมการแล้วจะทำให้ f(E) = 0

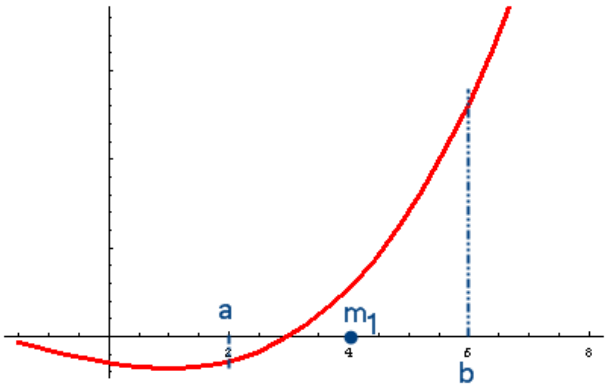
การหารากสมการโดยวิธีแบ่งครึ่งช่วง เริ่มต้น ด้วยการระบุช่วงตัวเลขที่ต้องการหารากสมการขึ้นมาก่อน สมมติว่าอยู่ในช่วง a ถึง b โดยที่ค่า b มีค่ามากกว่า a เขียนเป็นสัญลักษณ์ได้เป็น [a,b] a คือ ขีดจำกัดล่าง b คือ ขีดจำกัดบน ค่าฟังก์ชัน f(x) ในช่วง [a,b] นี้ต้องมีค่าต่อเนื่อง

เมื่อนำ a และ b แทนค่าลงในฟังก์ชันแล้ว ผลคูณของค่าฟังก์ชันจะต้องมีค่าเป็นลบ ($f(a).f(b) < 0$)

ขั้นตอนการหารากสมการมีดังนี้

ขั้นที่ 1 เลือกค่า a และ b โดยการสุ่มแต่ต้องทำให้$f(a).f(b)$ น้อยกว่า ศูนย์ หรือเส้นกราฟมีจุดตัดบนแกน x

ขั้นที่ 2 หาค่ารากสมการโดยประมาณโดยแบ่งครึ่งช่วง ระหว่าง a กับ b ให้จุดที่แบ่งครึ่ง (ครั้งที่ 1) นี้คือ m_1 (m คือ midpoint) $m_1 = (a + b)/2$ ดังรูป 2.6



รูปที่ 2.6 การแบ่งครึ่งช่วงครั้งที่ 1

ขั้นที่ 3 ขณะนี้ช่วง a และ b จะถูกแบ่งเป็น 2 ช่วง โดยมี m_1 เป็นจุดแบ่งครึ่ง ตรวจสอบดูว่ารากสมการที่แท้จริงอยู่ช่วงใด

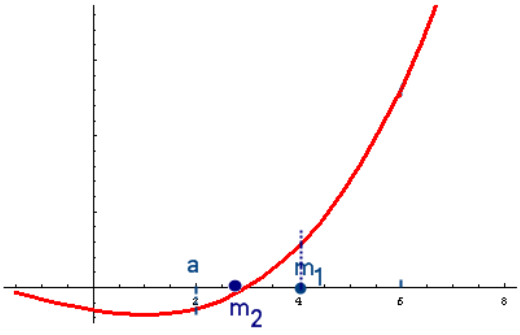
ก. ถ้า $f(a).f(m_1) < 0$ แสดงว่ารากสมการอยู่ในช่วง $[a, m_1]$

ข. ถ้า $f(m_1).f(b) < 0$ แสดงว่ารากสมการอยู่ในช่วง $[m_1 , b]$

ค. ถ้า $f(a).f(m_1) = 0$ หรือน้อยกว่าความคลาดเคลื่อนที่ยอมรับได้แสดงว่า m_1 คือรากสมการและสิ้นสุดการคำนวณ

ขั้นที่ 4 ถ้า m_1 ยังไม่ใช่รากสมการที่ต้องการ ให้แบ่งครึ่งช่วงอีก (ย้อนกลับไปขั้นที่ 2) จากรูปที่ 2.7

$m_2 = (a + m_1)/2$



รูปที่ 2.7 การแบ่งครึ่งช่วงในขั้นตอนที่ 2

กระทำซ้ำเช่นนี้เรื่อย ๆ จนถึง k ครั้ง เมื่อได้ $f(m_{k-1}).f(m_k)$ เท่ากับศูนย์หรือน้อยกว่าความคลาดเคลื่อนที่กำหนดแล้ว ค่า m_k นี้คือรากสมการที่ต้องการ

การแบ่งครึ่งช่วงแต่ละครั้ง จะทำให้ความกว้างของช่วงตัวเลขแคบลงเรื่อย ๆ แต่ละครั้งจุด m_k จะเป็นค่าประมาณของรากสมการ การกระทำซ้ำจะสิ้นสุดลงเมื่อ m_k อยู่ในช่วงความคลาดเคลื่อนที่ยอมรับได้ (given tolerance)

เพื่อป้องกันความผิดพลาดที่ไม่คาดคิดเกิดขึ้น ทำให้โปรแกรมทำงานวนรอบไม่สิ้นสุด จึงกำหนดจำนวนรอบสูงสุดของการกระทำซ้ำไว้ด้วย รายละเอียดของโปรแกรมที่ใช้คำนวณหารากสมการโดยวิธีแบ่งครึ่งช่วง ได้แสดงไว้ในภาคผนวก 3

2.3.3 การหาปริพันธ์โดยใช้วิธีแบ่งเป็นสี่เหลี่ยมคางหมู (Trapezoidal rule)

การหาปริพันธ์ใช้ในการ normalize ฟังก์ชันคลื่น ตามสมการ (2.5) ในที่นี้เลือกใช้วิธีแบ่งพื้นที่ใต้เส้นโค้งให้เป็นรูปสี่เหลี่ยมคางหมูเล็ก ๆ เป็นจำนวนมาก แล้วรวมพื้นที่เล็ก ๆ ของสี่เหลี่ยมคางหมูเหล่านี้จะได้เป็นผลลัพธ์ของการหาปริพันธ์ในช่วงที่ต้องการ

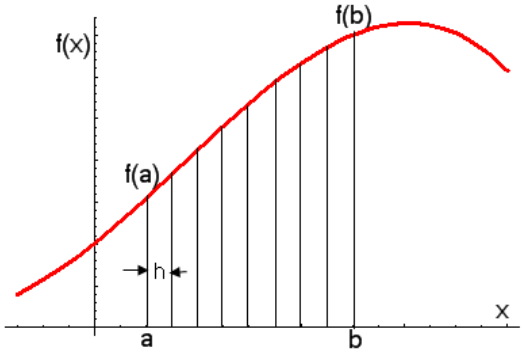
การหาปริพันธ์ของฟังก์ชัน $f(x)$ ที่มีค่าต่อเนื่องในช่วง a, b โดยที่ $a \leq x \leq b$ เขียนเป็นสัญลักษณ์ $\int_a^b f(x)dx$ มีค่าเท่ากับพื้นที่ใต้เส้นโค้งของ $f(x)$ ในช่วงดังกล่าว

การหาพื้นที่ใต้เส้นโค้ง ทำได้โดยแบ่งพื้นที่ออกเป็นรูปเปลี่ยมเล็ก ๆ จำนวนมาก (เรียกรูปเหลี่ยมเล็ก ๆ นี้ว่า quadrilaterals หรือ numerical quadrature) จากนั้นรวมพื้นที่เล็ก ๆ เหล่านี้เข้าด้วยกัน กลายเป็นพื้นที่ส่วนที่แรเงาทั้งหมด

เขียนเป็นสมการได้ดังนี้

$$\int_a^b f(x)dx \qquad = \qquad \sum_{i=1}^{N} f(x_i)\,\omega_i$$

เมื่อ N คือจำนวนเส้นที่แบ่งพื้นที่ใต้โค้งจากรูป N = 9 จะแบ่งพื้นที่ออกเป็น 8 ส่วนเล็ก ๆ ω_i คือค่าน้ำหนัก (weight) ของ $f(x)$ ของแต่ละค่า i



รูปที่ 2.8 แสดงการแบ่งพื้นที่ในช่วง x =a ถึง x = b เป็นสี่เหลี่ยมคางหมูเล็ก ๆ

แบ่งพื้นที่ใต้โค้งในช่วง a ถึง b ออกเป็นช่องเล็ก (interval) กว้างช่องละ h แต่ละช่องลากเส้นตรงเชื่อมจุดบนเส้นโค้งจะเกิดเป็นสี่เหลี่ยมคางหมูเป็นจำนวนมาก รวมสี่เหลี่ยมคางหมูเล็ก ๆ เหล่านี้ทั้งหมดจะได้เป็นค่าประมาณของการหาปริพันธ์ของ f(x) ในช่วง a ถึง b ยิ่งแบ่งช่วงระยะ h ให้มีค่าน้อยเท่าใดเส้นตรงที่ลากเชื่อมจุดต่าง ๆ บนเส้นโค้งจะทับกับเส้นกราฟของฟังก์ชันที่แท้จริงมากขึ้นเพียงนั้น

ที่จุด x_i ใด ๆ สามารถหาค่าโดยวัดจาก a เป็นหลัก

$$x_i = a + (i-1)h \quad \text{เมื่อ} \quad I = 1, 2, \ldots N$$

เมื่อแทน $x_I = b$ โดยที่ $i = N$ จะได้

$$h = \frac{b-a}{N-1}$$

พื้นที่สี่เหลี่ยมคางหมูแต่ละส่วนเล็ก ๆ

$$\int_{x_i}^{x_i+h} f(x)dx \cong \frac{1}{2} \times \text{ฐาน} \times (\text{ผลบวกด้านคู่ขนาน})$$

$$= \frac{1}{2} h\,(f(x_i) + f(x_{i+1}))$$

พื้นที่ทั้งหมดใต้เส้นโค้งในช่วง [a, b] คือ

$$\int_a^b f(x)dx \cong \frac{h}{2} f(x_1) + hf(x_2) + hf(x_3) + \ldots + hf(x_{N-1}) + \frac{h}{2} f(x_N)$$

$$= \left(\sum_{i=1}^{n} f(x_i) - \frac{f(x_1)}{2} - \frac{f(x_n)}{2} \right) h$$

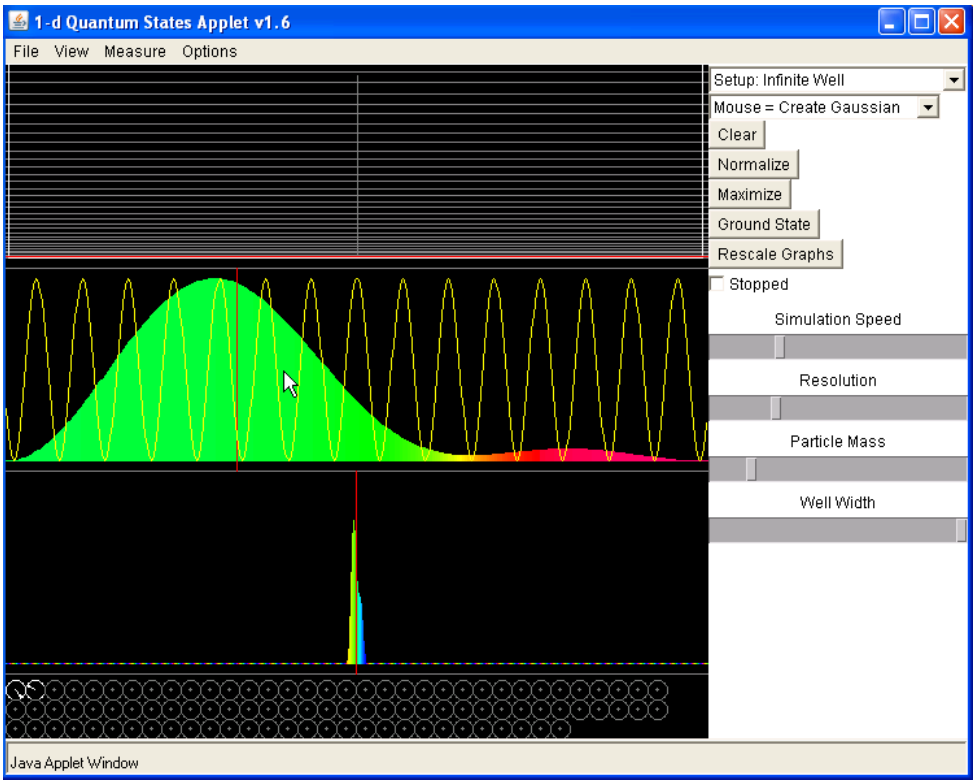
จุด x_i ที่ไม่ใช่เป็นจุดปลายจะถูกนับ 2 ครั้ง จากสมการจะเห็นว่าค่าน้ำหนัก (w_i) ที่ x_i ใด ๆ คือ

$$w_i = \left\{ \frac{h}{2}, h, \ldots, h, \frac{h}{2} \right\}, \; i=1, N$$

2.4 งานวิจัยหรือโปรแกรมเกี่ยวกับการจำลองสถานการณ์ในกลศาสตร์ควอนตัม

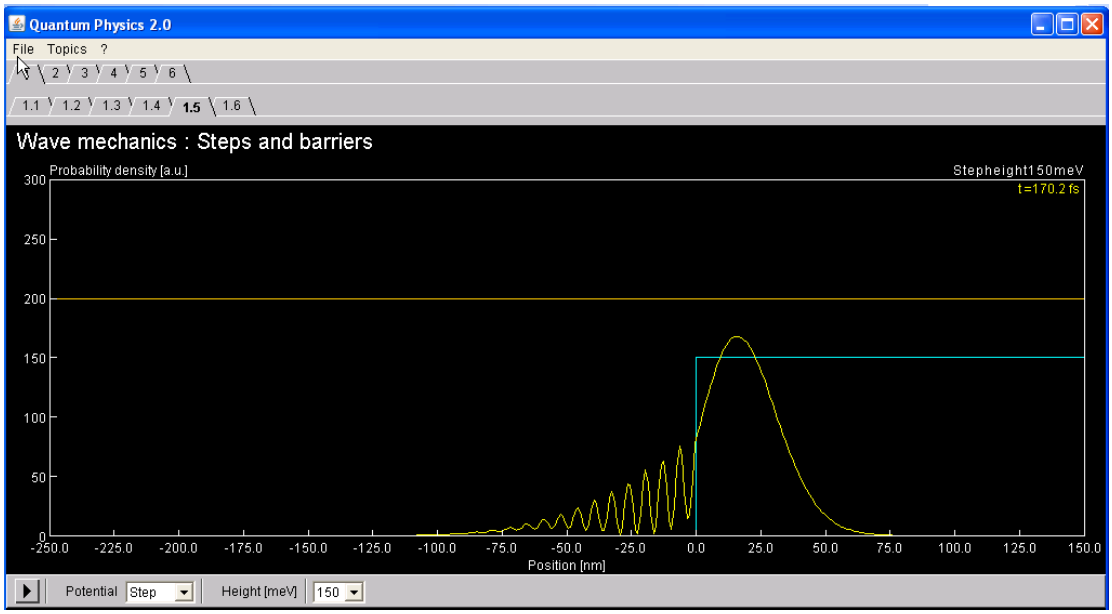
การจำลองสถานการณ์ในกลศาสตร์ควอนตัม ได้มีผู้จัดทำและเผยแพร่ ผลงานเป็นที่น่าสนใจมีดังนี้

Falstad, Paul ได้สร้างโปรแกรมแสดงการเคลื่อนที่ของอนุภาคแบบ 1 มิติ ในบ่อศักย์แบบสี่เหลี่ยม แสดงผลในรูปของ applet สามารถเข้าถึงได้จาก http://www.falstad.com/



รูปที่ 2.9 แสดงการจำลองอนุภาคในบ่อศักย์แบบสี่เหลี่ยม

Joffre, M. Basdevant, J. และ Dalibard, J. ภาควิชาฟิสิกส์ สถาบัน Ecole Polytechnique ประเทศฝรั่งเศส เข้าถึงโดย http://www.quantum-physics.polytechnique.fr/ ได้สร้างโปรแกรมจำลองการแพร่ของ Gaussian wave packet เมื่อคลื่นกระทบกับกำแพงศักย์ และศักย์ที่มีค่าเป็นขั้น โดยใช้ค่าเริ่มต้นได้มาจากการทดลองให้อิเล็กตรอนเคลื่อนที่ในสารกึ่งตัวนำ GaAs กำหนดพลังงานจลน์ของอิเล็กตรอนมีค่าเริ่มต้นที่ 200 eV เมื่อกำหนดให้กำแพงศักย์มีค่าสูงกว่าพลังงานของอิเล็กตรอนมาก ๆ จะเห็นการสะท้อนของคลื่นได้อย่างชัดเจน ตรงกับคำทำนายของกลศาสตร์คลาสสิก และเกิดปรากฏการณ์แทรกสอดระหว่างคลื่นตกกระทบกับคลื่นสะท้อนให้เห็นด้วย มีคลื่นบางส่วนที่สามารถส่งผ่านไปยังกำแพงศักย์ ซึ่งจะมีความเร็วลดลงมาก เนื่องจากพลังงานจลน์ของอิเล็กตรอนลดค่าลง เป็นการแสดงเหตุการณ์การเกิด Tunnel effect

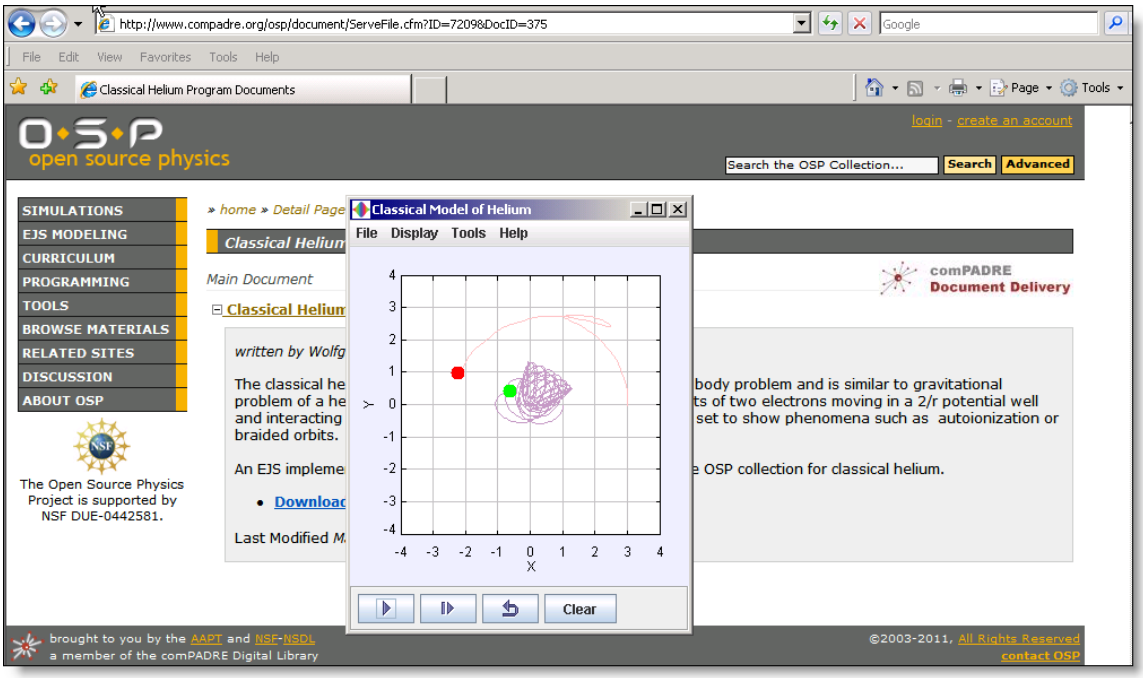


รูปที่ 2.10 แสดงสมบัติความเป็นคลื่นของอนุภาค

Raedt H. ศูนย์ฟิสิกส์ทฤษฎีและวัสดุศาสตร์ (Theoretical Physics and Materials Science Centre) ของมหาวิทยาลัย Groningen เว็บไซต์ http://rugth30.phys.rug.nl/quantummechanics/ ได้สร้างบทเรียนวิชากลศาสตร์ควอนตัม จำลองสถานการณ์ต่าง ๆ ในบทเรียน เช่น การเคลื่อนที่ของอนุภาคอย่างอิสระ การเคลื่อนที่ของอนุภาคในสนามของแรง บางส่วนเขียนด้วยภาษา FORTRAN บางส่วนเขียนด้วย FLASH จากนั้นจับภาพเคลื่อนไหวเหล่านั้นแปลงเป็น video file แสดงภาพเคลื่อนไหวโดยใช้โปรแกรม real player

Open Source Physics หรือ OSP มีเว็บไซต์อยู่ที่ http://www.compadre.org/osp/ เป็นโครงการที่ได้รับการสนับสนุนจาก National Science Foundation and วิทยาลัยเดวิสัน (Davidson College) เน้นการศึกษาฟิสิกส์โดยใช้คอมพิวเตอร์เข้ามามีส่วนช่วยในการเรียนการสอน ได้จัดทำคลังโปรแกรมที่เปิดเผยโปรแกรมต้นฉบับ (open source code libraries) เกี่ยวกับการคำนวณเชิงตัวเลข โปรแกรมจำลองปรากฏการณ์ทางฟิสิกส์ด้านต่าง ๆ จัดทำเครื่องมือที่ช่วยทำให้การเขียนโปรแกรมคอมพิวเตอร์สำหรับเรียนรู้บทเรียนทางฟิสิกส์ได้ง่ายขึ้นเรียกว่า Ejs หรือ Easy java simulation มีทำบทเรียนชุดสำเร็จสำหรับฟิสิกส์ขั้นสูง ในชุดบทเรียนประกอบด้วยหนังสือประกอบการเรียน โปรแกรมคอมพิวเตอร์ที่เกี่ยวข้องกับเรื่องนั้น ๆ และใบงานที่ใช้ในบทเรียน บทเรียนบางเรื่องสามารถศึกษาและดาวน์โหลดได้ผ่านทางเว็บไซต์

โปรแกรมทั้งหมดเขียนเป็นภาษาจาวามีการเปิดเผยรายละเอียดของ source code ทั้งหมด สามารถนำไปพัฒนาต่อยอด หรือดัดแปลงแก้ไขใช้ร่วมกับโปรแกรมที่กำลังพัฒนา



รูปที่ 2.11 เว็บไซต์ของ Open source physics

บทที่ 3
วิธีการดำเนินการวิจัย

แบบจำลองอนุภาคในบ่อศักย์แบบต่าง ๆ ในกลศาสตร์ควอนตัม เป็นการพัฒนาโปรแกรมคอมพิวเตอร์เพื่อใช้จำลองสถานการณ์กรณีที่อนุภาคอยู่ภายในบ่อศักย์ ศึกษาพลังงานของอนุภาคซึ่งพบว่ามีค่าไม่ต่อเนื่อง ขนาดของพลังงานขึ้นอยู่กับเลขควอนตัมของระบบ แสดงฟังก์ชันคลื่นของอนุภาคที่อยู่ในบ่อศักย์นั้นที่เลขควอนตัมต่าง ๆ ฟังก์ชันคลื่นที่ได้จะนำไปสู่ความน่าจะเป็นที่จะพบอนุภาคภายในบ่อศักย์นั้น วิธีการดำเนินการวิจัย แบ่งเป็นหัวข้อดังนี้

3.1 เครื่องมือที่ใช้ในการวิจัย

3.2 ขั้นตอนการพัฒนาโปรแกรม

3.3 โครงสร้างของแฟ้มข้อมูล (Folder) ที่ใช้ในการเขียนโปรแกรม

3.4 การแปลโปรแกรมให้เป็น class file และบีบอัดให้เป็น Jar file

3.1 เครื่องมือที่ใช้ในการวิจัย

เครื่องมือที่ใช้ในการทำวิจัยซึ่งมีอยู่แล้ว ไม่จำเป็นต้องจัดหาเพิ่มเติม ดังต่อไปนี้

1. เครื่องคอมพิวเตอร์สำหรับใช้เขียนโปรแกรม และทดสอบโปรแกรม

2. ซอฟต์แวร์ Java SDK 6 (java Standard Development Kit) สามารถ download ได้โดยไม่เสียค่าใช้จ่ายจาก http:/java./sun.com ติดตั้งใช้งานภายใต้ระบบปฏิบัติการวินโดว์ (ในที่นี้ใช้ Windows XP service pack 3)

3. ซอฟต์แวร์ที่ใช้เป็น editor สำหรับเขียน source code ของโปรแกรม

4. ซอฟต์แวร์ที่ใช้ในการตรวจสอบความถูกต้องของผลการคำนวณของโปรแกรม ในที่นี้จะใช้ MathematicA version 5.1 และโปรแกรมตารางคำนวณ Excel (Microsoft Excel version 2003) ช่วยในการพล็อตกราฟของฟังก์ชันคลื่น ในช่วงที่เขียนโปรแกรมในลักษณะ text mode

5. เครื่องคอมพิวเตอร์ทำหน้าที่เป็น server หมายเลข IP คือ 203.158.100.100 ใช้โปรแกรม IIS 7 (Internet Information System version 7) เป็น web server ภายใต้ระบบปฏิบัติการวินโดว์ 2008 สำหรับทดสอบการทำงานของโปรแกรมผ่านเครือข่ายอินเทอร์เน็ต

3.2 ขั้นตอนการพัฒนาโปรแกรม

ในการพัฒนาโปรแกรมคอมพิวเตอร์ เพื่อจำลองพฤติกรรมของอนุภาคในบ่อศักย์นั้น ได้แบ่งขั้นตอนการพัฒนาเป็นลำดับขั้นดังนี้

1. ประมวลความรู้และทฤษฎีที่เกี่ยวข้องกับสมการชเรอดิงเงอร์ โดยเน้นไปที่กรณีสมการไม่ขึ้นกับเวลา (time independent) ใน 1 มิติ ศึกษาการแก้ปัญหาเชิงวิเคราะห์เมื่ออนุภาคที่ถูกจำกัดบริเวณในบ่อศักย์แบบต่าง ๆ (Bound state) รายละเอียดของเนื้อหา ดังปรากฏในบทที่ 2 ทฤษฎีและเอกสารที่เกี่ยวข้อง

2. ศึกษาและออกแบบระเบียบวิธีการคำนวณเชิงตัวเลขที่ต้องใช้ในการหาผลเฉลยของสมการชเรอดิงเงอร์ ซึ่งเป็นสมการอนุพันธ์อันดับสอง ระเบียบวิธีที่ถูกเลือกใช้ในงานวิจัยชิ้นนี้คือ วิธีของ นูเมอรอฟ (Numerov) ใช้ในการหาค่าฟังก์ชันคลื่นของอนุภาคที่กำหนดค่าพลังงานมาให้ การหาปริพันธ์ของฟังก์ชันซึ่งจะนำไปใช้ในการ normalize ฟังก์ชันคลื่น โดยใช้วิธีแบ่งพื้นที่เป็นสี่เหลี่ยมคางหมูเล็ก ๆ และรวมพื้นที่ทั้งหมดเข้าด้วยกัน การหารากสมการของฟังก์ชันใช้วิธีแบ่งครึ่งช่วง (Bisection) ตรวจสอบการคำนวณพลังงานของอนุภาคที่ได้จากโปรแกรมคอมพิวเตอร์ มีค่าใกล้เคียงกับค่าที่ได้จากการวิเคราะห์หรือไม่

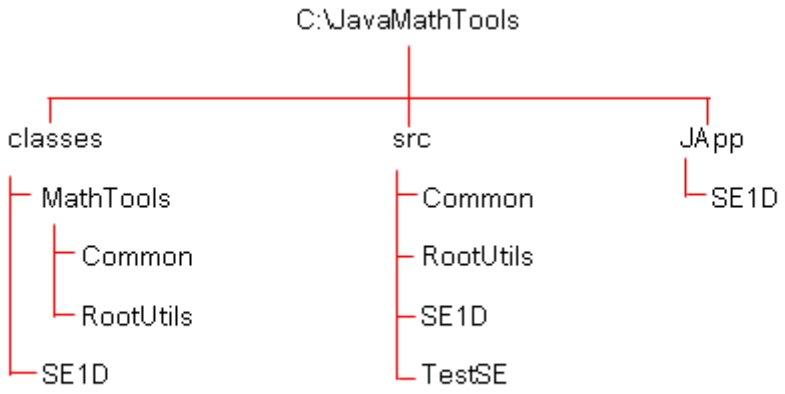
3. ออกแบบโปรแกรมและเขียนคำสั่งเป็นภาษาจาวา (Java) โดยแบ่งเป็น 2 ช่วง ช่วงแรกจะให้โปรแกรมแสดงผลเป็นแบบ text mode เพื่อตรวจสอบความถูกต้องของผลลัพธ์ที่ได้จากการคำนวณว่าสอดคล้องกับทฤษฎี เมื่อไม่พบข้อผิดพลาดในการคำนวณแล้ว จึงดำเนินการช่วงที่ 2 โดยนำผลลัพธ์ที่ได้นั้นเป็นข้อมูลในการแสดงผลทางด้านกราฟิก เลือกใช้ GUI (Graphic User Interface) แบบ SWING ซึ่งมีการจัดการวางรูปแบบ (Layout Manager) ที่นำไปใช้งานได้ง่าย มีความยืดหยุ่นในการวางองค์ประกอบ เหมาะสมกับการใช้งานผ่าน เครือข่ายอินเทอร์เน็ต

4. ทดสอบการทำงานของโปรแกรมจำลองอนุภาคในบ่อศักย์ ในภาพรวมทั้งหมด ปรับปรุงและแก้ไขชุดคำสั่ง ในกรณีที่ประมวลผลแล้วคลาดเคลื่อน หรือมีความบกพร่องในการแสดงผลทางกราฟิก

5. นำไปประกอบกับบทเรียน E-learning ที่เกี่ยวกับกลศาสตร์ควอนตัม เช่นในวิชากลศาสตร์ควอนตัม 1 ที่ใช้สอนนักศึกษาของสาขาวิชาฟิสิกส์ หรือวิชาฟิสิกส์ 2 ในหัวข้อ กลศาสตร์ควอนตัมเบื้องต้น ใช้สอนนักศึกษาคณะวิศวกรรมศาสตร์ หรือวิชาฟิสิกส์ยุคใหม่ ในหัวข้อกลศาสตร์ควอนตัม ใช้สอนนักศึกษาของคณะครุศาสตร์และนักศึกษาที่เรียนสาขาฟิสิกส์ นำบทเรียนเหล่านี้ไปติดตั้งไว้ใน Web server ของสาขาวิชาฟิสิกส์ (http://www.electron.rmutphysics.com/SE1D)

3.3 โครงสร้างของแฟ้มข้อมูล (Folder) ที่ใช้ในการเขียนโปรแกรม

เพื่อความเป็นระเบียบและเป็นระบบในการพัฒนาโปรแกรม และเกิดความสะดวกในการนำไปใช้งาน และแก้ไขปรับปรุงโปรแกรมในอนาคต ผู้วิจัยจึงได้จัดสร้างแฟ้มข้อมูล มีลักษณะเป็นลำดับชั้นดังนี้ (แสดงเฉพาะแฟ้มข้อมูลที่เกี่ยวข้องกับการพัฒนาโปรแกรมแบบจำลองอนุภาคในบ่อศักย์เท่านั้น)



ภายใต้โฟลเดอร์ C:\JavaMathTools จะมีโฟลเดอร์ย่อยที่สำคัญ 3 โฟลเดอร์คือ

โฟลเดอร์ classes จะใช้เก็บ class file ที่ได้จากการแปลภาษาจาวาเป็น byte code ในโฟลเดอร์นี้จะแยกออกเป็น class ที่เกี่ยวกับการคำนวณเชิงตัวเลข จะเก็บไว้ในโฟลเดอร์ MathTools ซึ่งจะแยกไปตามหัวข้อที่เกี่ยวข้องกับการคำนวณเชิงตัวเลข ค่าคงที่หรือตัวแปร รวมทั้งคลาสที่ต้องใช้ร่วมกันจะถูกเก็บไว้ใน Common การหารากสมการของฟังก์ชันใด ๆ จะถูกเก็บไว้ใน RootUtils ส่วนการคำนวณฟังก์ชันคลื่น และระดับพลังงานค่าต่าง ๆ หรือการคำนวณใดที่เกี่ยวข้องกับสมการชเรอดิงเงอร์ จะถูกเก็บไว้ใน SE1D (ย่อมาจาก Schrodinger Equation 1 Dimension)

โฟลเดอร์ src จะเก็บไฟล์ต้นฉบับ (source code) ที่มีนามสกุลไฟล์เป็น .java การเก็บไฟล์ต้นฉบับจะเก็บในโฟลเดอร์ย่อยที่มีชื่อเหมือนกับโฟลเดอร์ classes โฟลเดอร์ที่เพิ่มขึ้นมาคือ TestSE จะเก็บไฟล์ต้นฉบับที่ใช้ในการทดสอบโปรแกรมที่อยู่ในโฟลเดอร์ SE1D หรือเป็นไฟล์ต้นฉบับที่เรียกใช้งานคลาสอื่น ๆ ผสมกับคลาสที่อยู่ในโฟลเดอร์ SE1D ด้วยเช่นกัน ในโฟลเดอร์นี้อาจพบโฟลเดอร์ย่อย เรียงลำดับตั้งแต่ 01, 02, 03, หมายถึงการพัฒนาโปรแกรมที่มีการปรับปรุงเปลี่ยนแปลงเรียงลำดับตามรุ่นที่พัฒนา โดยที่ยังคงรักษาต้นฉบับของรุ่นก่อน ๆ ไว้ เผื่อไว้ในบางครั้งในการพัฒนาอาจได้ผลลัพธ์ที่ไม่ตรงกับจุดประสงค์ หรือโปรแกรมมีข้อบกพร่องหรือมีความคลาดเคลื่อนที่แก้ไขได้ยาก ผู้พัฒนาสามารถย้อนกลับไปยังจุดก่อนหน้านั้น โดยไม่ต้องเริ่มต้นที่ศูนย์ใหม่

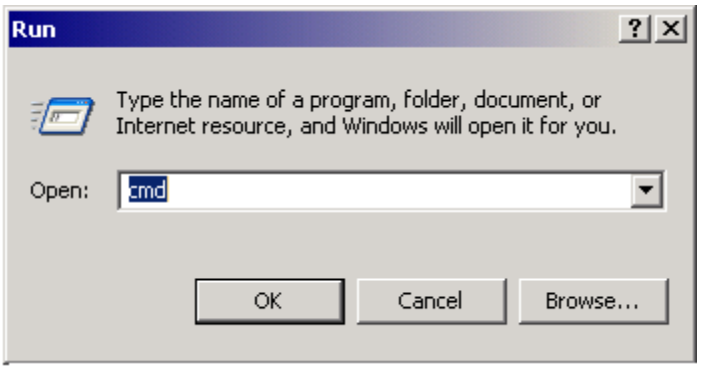
โฟลเดอร์ JApp จะเก็บ class files ที่ได้จากการแปลโปรแกรมต้นฉบับที่เก็บไว้โฟลเดอร์ TestSE ส่วนที่เป็น main ของโปรแกรมจะถูกเก็บไว้ในโฟลเดอร์นี้

3.4 การแปลโปรแกรมให้เป็น class file และบีบอัดให้เป็น Jar file

ในการแปลโปรแกรมหรือ compile โปรแกรมต้นฉบับ (ที่มีนามสกุล .java) ให้เป็น class file และถูกจัดเก็บไว้ในโฟลเดอร์ที่เหมาะสมนั้น จะต้องใช้คำสั่ง และ option ในการแปลโปรแกรมให้ถูกต้องด้วย มิฉะนั้นผลที่ตามมาคือโปรแกรมไม่สามารถทำงานได้

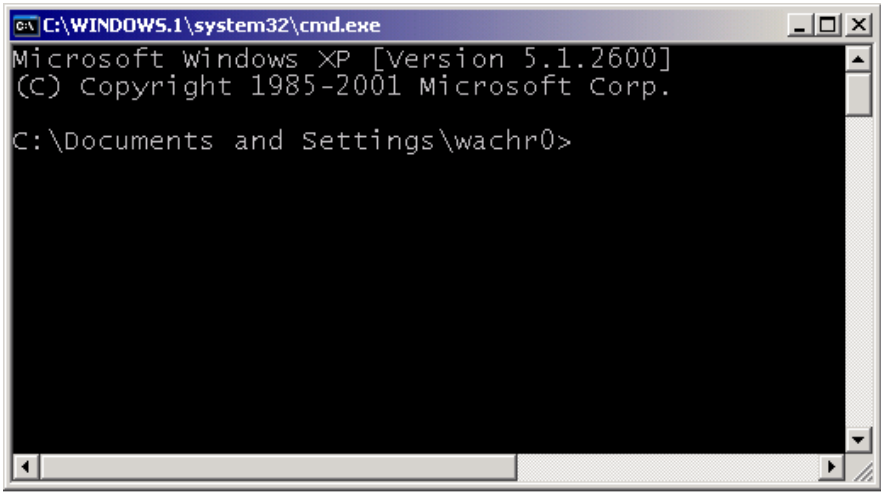
ในการคอมไพล์โปรแกรมเริ่มต้นดังนี้

1. คลิกที่ปุ่ม Start และคลิกที่คำสั่ง run แล้วพิมพ์คำสั่ง cmd ดังรูป เพื่อเข้าสู่ DOS mode หรือ Command mode



รูปที่ 3.1 การเข้าสู่ DOS mode

2. เมื่อคลิกที่ปุ่ม OK จะได้หน้าต่าง command mode



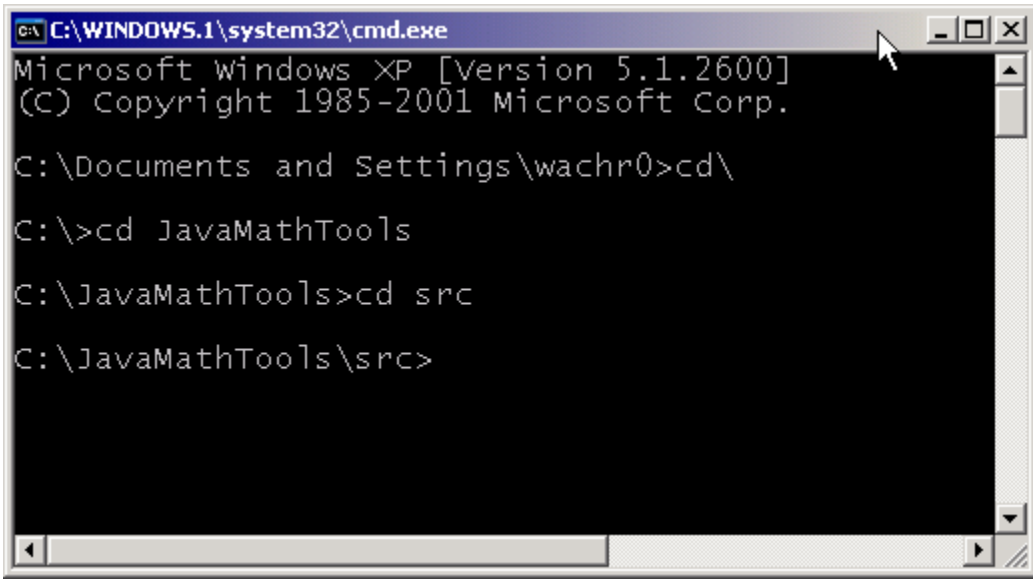
รูปที่ 3.2 แสดงหน้าต่าง เมื่อเข้าสู่ DOS mode

3. พิมพ์คำสั่ง cd (ย่อมาจาก change directory) ดังปรากฏในภาพ เพื่อเข้าไปอยู่ในโฟลเดอร์ JavaMathTools/src

ความหมายของคำสั่งมีดังนี้

พิมพ์คำสั่ง cd \ (cd ตามด้วยเครื่องหมาย back slash) เป็นคำสั่งที่ให้กลับไปที่โฟลเดอร์ราก คือ c:\

คำสั่ง cd JavaMathTools และ cd src เป็นการสั่งให้ไปที่โฟลเดอร์ย่อย JavaMathTools และโฟลเดอร์ย่อยลงไปอีกชื่อ src

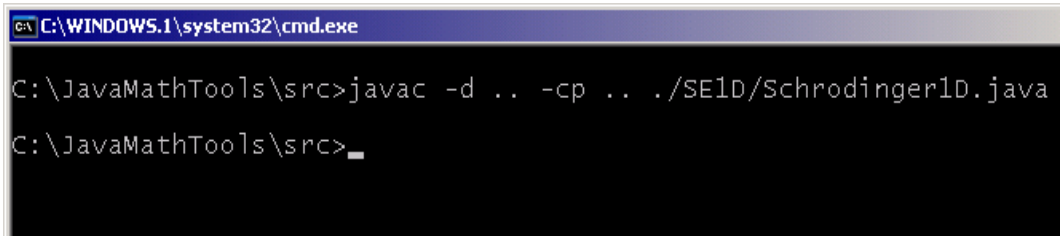


รูปที่ 3.3 แสดงโฟลเดอร์ที่เก็บ source code

4. สมมติว่าจะคอมไพล์โปรแกรมชื่อ Schrodinger1D.java ซึ่งถูกเก็บไว้ใน โฟลเดอร์ src/SE1D ให้ใช้คำสั่งดังนี้

javac –d .. –cp .. ./SE1D/Schrodinger1D.java กดปุ่ม Enter

ถ้าโปรแกรมต้นฉบับไม่มีข้อผิดพลาด หน้าจอจะไม่แสดงข้อผิดพลาดใด ๆ จะเป็นดังรูป

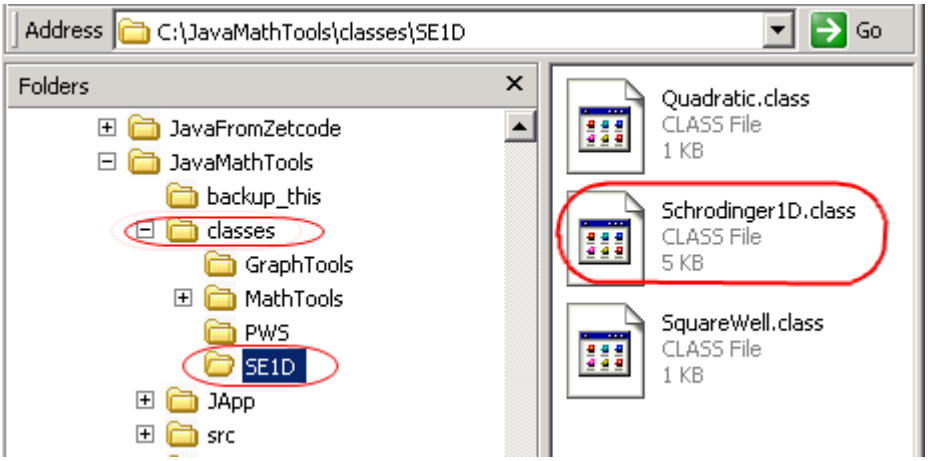


รูปที่ 3.4 เมื่อใช้คำสั่ง javac คอมไพล์โปรแกรม

Option ที่ปรากฏท้ายคำสั่ง javac คือ –d หมายถึง โฟลเดอร์ที่จะเก็บโปรแกรมซึ่งมีนำสกุลเป็น class ในที่นี้เป็น จุด 2 จุด หมายถึงให้เก็บ class file ที่ได้จากการคอมไพล์นี้ไว้ในโฟลเดอร์ที่อยู่เหนือ src ขึ้นไป 1 ลำดับชั้น

–cp นั้นย่อมาจากคำว่า class path เป็นการระบุว่า class file ที่เป็น library นั้น ตัวคอมไพเลอร์สามารถไปค้นหาโดยเริ่มต้นจากโฟลเดอร์ที่อยู่เหนือ src ไป 1 ลำดับชั้นเช่นเดียวกัน ./SE1D/Schrodinger1D.java เป็นการระบุชื่อไฟล์ที่จะทำการคอมไพล์

ไฟล์ Schrodinger1D.class ที่ผ่านการคอมไพล์แล้วจะถูกเก็บไว้ในโฟลเดอร์ classes/SE1D/ ดังรูป



รูปที่ 3.5 แสดงโฟลเดอร์ที่เก็บ class file

การคอมไพล์ไฟล์ต้นฉบับอื่น ๆ ก็ทำเช่นเดียวกัน เพียงแต่ระบุโฟลเดอร์ของไฟล์ต้นฉบับให้ถูกต้องตามที่เป็นจริงเท่านั้น เช่นการคอมไพล์ ไฟล์ชื่อ Common.java ใช้คำสั่งดังนี้

javac –d .. –cp .. ./Common/Common.java

หรือจะคอมไพล์ไฟล์ Biseciton.java ใช้คำสั่งดังนี้

javac –d .. –cp .. ./RootUtils/Bisection.java

การสร้าง Jar file

ไฟล์นามสกุล .class ที่เกิดจากการคอมไพล์จะมีเป็นจำนวนมาก อาจไม่สะดวกในการขนย้าย เมื่อนำไปใช้กับเครื่องอื่น ๆ หรือใช้งานในระบบเครือข่าย การนำคลาสไฟล์ที่เกี่ยวข้องทั้งหมดมารวบรวมและบีบอัดให้มีขนาดไฟล์เล็กลง สามารถขนย้ายโดยใช้ไฟล์นี้เพียงไฟล์เดียว เรียกไฟล์นี้ว่า Jar file ย่อมาจากคำว่า **J**ava **Ar**chive file การรวบรวมคลาสไฟล์ให้เป็น Jar file ทำได้ดังนี้

1. สร้าง Manifest file ชื่อ SEManifest.txt เก็บไว้ใน c:\JavaMathTools โดยใช้ editor (เช่น Notepad) พิมพ์ข้อความต่อไปนี้

Main–Class: JApp.SE1D.mainApp

แล้ว save เก็บไว้ในชื่อ SEManifest.txt

2. ในการสร้างไฟล์ SE1D.jar ใช้คำสั่งดังนี้

C:\JavaMathTools>jar –cvmf SEManifest.txt SE1D.jar JApp/SE1D classes/SE1D classes/MathTools/Common

รูปที่ 3.6 การใช้คำสั่ง Jar

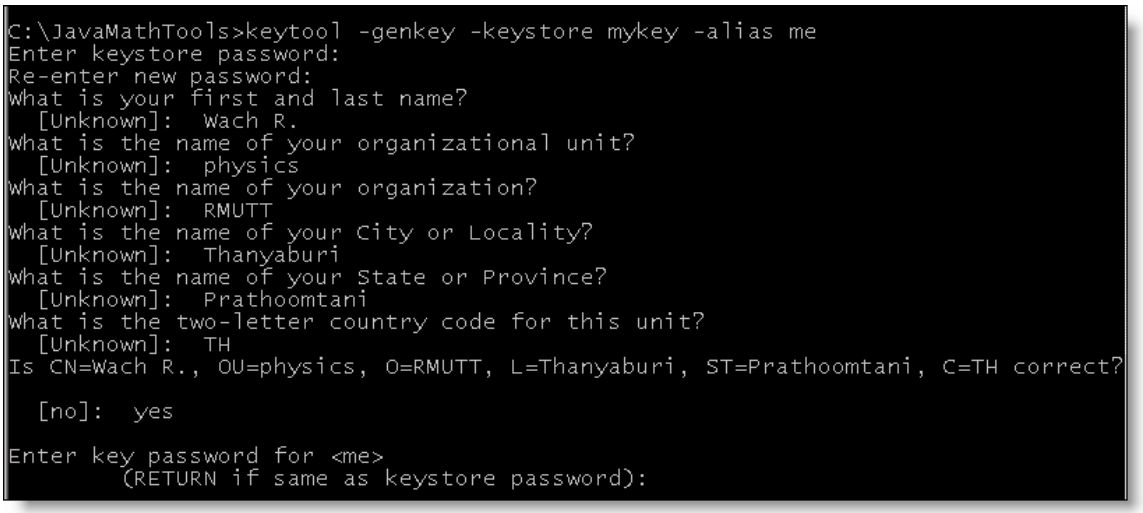
การนำ jar file ไปไว้ในแม่ข่ายอินเตอร์เน็ต จำต้องสร้างลายเซ็นสำหรับ jar file นี้ก่อน จุดประสงค์ของลายเซ็นนี้คือเป็นเครื่องหมายแสดงความปลอดภัยสำหรับการให้โปรแกรมนี้ทำงานผ่านเทคโนโลยี Java web start ถ้ามีการเปลี่ยนแปลงแก้ไขโปรแกรม (อาจเกิดจากไวรัส หรือผู้ไม่ประสงค์ดีนำคำสั่งมาเพิ่มเติม) หลังจากการสร้างลายเซ็นนี้แล้ว โปรแกรมจะถูกห้ามมิให้ทำงาน เพื่อป้องกันความเสียหายที่อาจจะเกิดขึ้นตามมาภายหลัง

สร้างลายเซ็น ตามลำดับดังนี้

1. สร้าง mykey สำหรับสร้าง self sign certificate เริ่มดังนี้

C:\JavaMathTools>keytool –genkey –keystore mykey – alias me

ใส่ Password ตามที่ต้องการ แต่ต้องจำไว้เพราะจะต้องนำไปใช้ในตอนท้าย ๆ ด้วย



รูปที่ 3.7 การสร้าง key สำหรับการเข้ารหัส jar file

2. สร้าง certificate สำหรับการ sign ไฟล์ SE1D.jar โดยใช้ keytool สร้าง self signed certificate

C:\JavaMathTools>keytools –selfcert –alias me –keystore mykey

ใส่ password ตามที่เคยใส่มาแล้วในข้อ 1

รูปที่ 3.8 การใช้ keytools สร้าง self certificate

3. sign ไฟล์ SE1D.jar โดยใช้คำสั่ง jarsigner

Passphrase ที่ใช้คือ ตัวเดียวกับที่ใช้มาแล้วข้างต้น

c:\JavaMathTools>jarsigner –keystore mykey SE1D.jar me

รูป 3.9 การสร้างลายเซ็นโดยใช้คำสั่ง jarsigner

บทที่ 4
ผลการวิจัย

ผลของการศึกษาแบบจำลองอนุภาคในบ่อศักย์แบบต่าง ๆ ซึ่งเขียนเป็นโปรแกรมภาษาจาวาแบ่งเป็นหัวข้อดังต่อไปนี้

4.1 ผลลัพธ์การจำลองการทำงานของโปรแกรมใน text mode

4.2 ผลลัพธ์การแสดงภาพกราฟิกของบ่อศักย์และฟังก์ชันคลื่นบนจอภาพ

4.3 โปรแกรมต้นฉบับ (source code)

4.1 ผลัพธ์การจำลองการทำงานของโปรแกรมใน text mode

การจำลองอนุภาคในบ่อศักย์ สามารถกำหนดให้โปรแกรมแสดงผลเฉพาะตัวอักษร (text mode) เพื่อตรวจสอบความถูกต้องและความคลาดเคลื่อนในการคำนวณของโปรแกรม สามารถแสดงผลได้รวดเร็วขึ้น เพราะหน่วยประมวลผลไม่ต้องไปเสียเวลากับการแสดงภาพกราฟิก

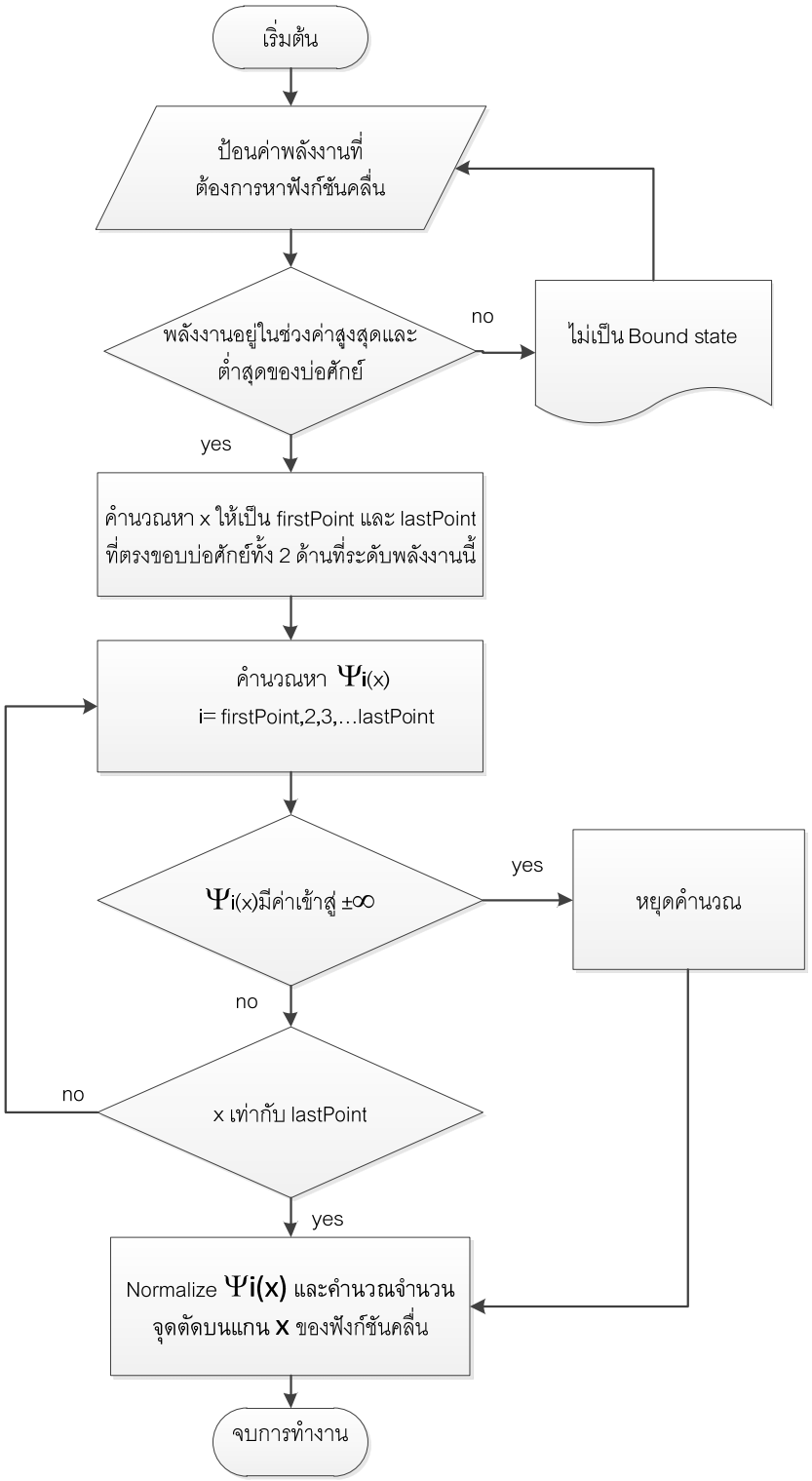
ไม่ว่าจะเป็นบ่อศักย์แบบใดสามารถใช้ระเบียบวิธี(algorithm) ต่อไปนี้คำนวณหาฟังก์ชันคลื่นและพลังงานของอนุภาคได้ทุกแบบ เพื่อให้ง่ายต่อความเข้าใจจึงแยกFlowchart เป็น 2 ส่วน ส่วนแรกเป็น Flowchart ของการหาฟังก์ชันคลื่น และ ส่วนที่ 2 เป็น Flowchart การหาพลังงานของอนุภาค ส่วนที่ 2 นี้ต้องใช้ขั้นตอนการหาฟังก์ชันคลื่นใน Flowchart ส่วนที่ 1 มาเกี่ยวข้องด้วย ในการจัดสร้างโปรแกรมได้รวมการหาฟังก์ชันคลื่น และการคำนวณพลังงานของอนุภาครวมไว้เป็นคลาสเดียวกัน ให้ชื่อคลาสว่า Schrodinger1D

การคำนวณหาฟังก์ชันคลื่นมีขั้นตอนดังต่อไปนี้

1. เลือกรูปแบบพลังงานศักย์ที่จะใช้ในการจำลองสถานการณ์
2. กำหนดขนาดของพลังงาน (E) ที่ต้องการหาฟังก์ชันคลื่น มีค่าไม่เกินค่าสูงสุดของบ่อศักย์ในช่วง x ที่กำหนดไว้
3. คำนวณหาผนังของบ่อศักย์ทั้งด้านซ้ายและด้านขวา ให้ x เป็นค่าแรก (first point) และค่าสุดท้าย (last point) ตรงจุดที่เป็นผนังของบ่อศักย์นี้
4. หาฟังก์ชันคลื่น $\psi(x_i)$ ทุก ๆ ค่าของ x_i i มีค่าตั้งแต่ first point จนถึง last point แต่ละค่า i ให้ตรวจสอบฟังก์ชันคลื่นว่าลู่ออกเกินกว่าค่าที่จำกัดไว้หรือไม่ ถ้าเกินให้หยุดการทำงาน
5. นับจำนวนจุดที่ฟังก์ชันคลื่นตัดกับแกน x
6. ทำการ normalize ฟังก์ชันคลื่นที่คำนวณได้

ขั้นตอนทั้งหมดสามารถเขียนเป็น Flow chart ได้ดังนี้

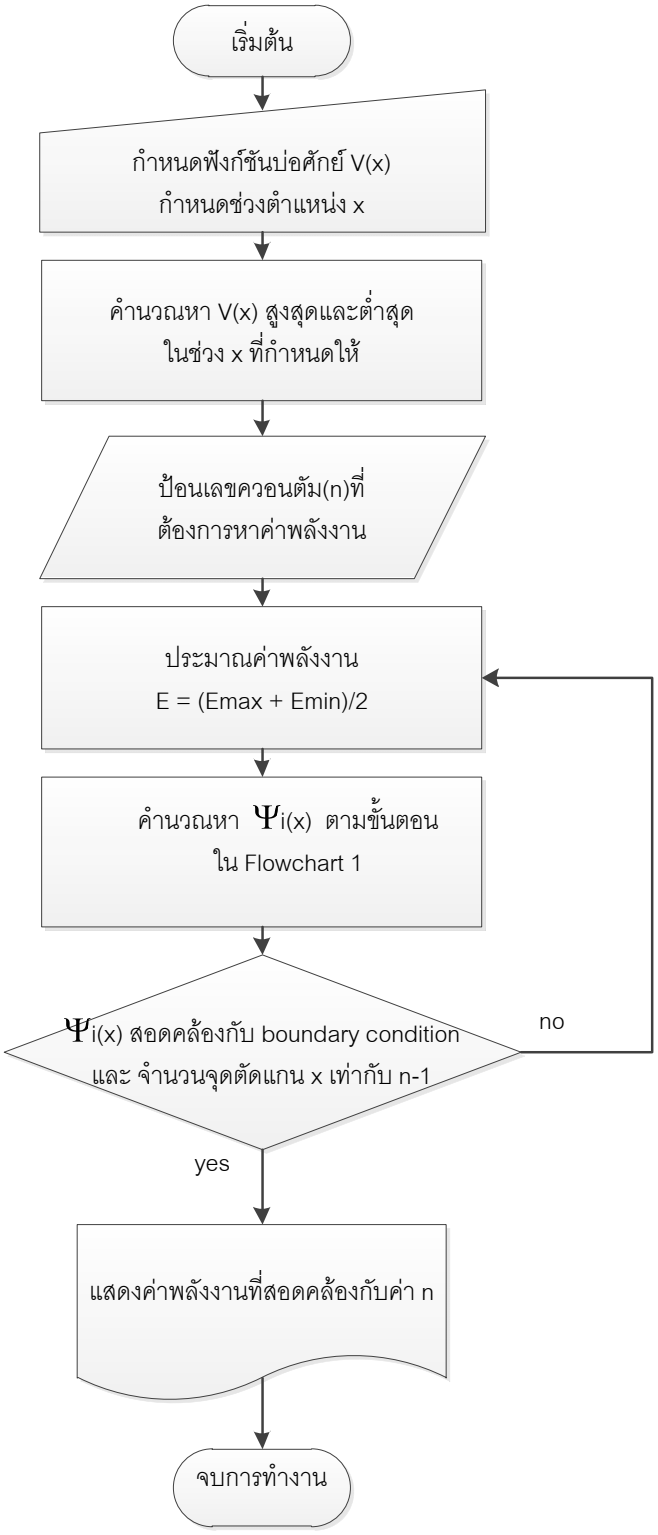
Flowchart 1: แสดงการหาฟังก์ชันคลื่นของอนุภาค
เมื่อกำหนดพลังงาน



การคำนวณหาพลังงานของอนุภาค (E) ซึ่งเป็นค่าเจาะจงของฟังก์ชันคลื่น มีขั้นตอนดังนี้

1. เลือกรูปแบบพลังงานศักย์ที่จะใช้ในการจำลองสถานการณ์
2. คำนวณพลังงานศักย์ค่าสูงสุดและต่ำสุดที่เป็นไปได้ในช่วง x ที่กำหนด
3. กำหนดระดับพลังงาน หรือเลขควอนตัม n ที่ต้องการหาพลังงาน
4. เริ่มต้นสุ่มค่าพลังงานที่ระดับพลังงาน n นี้ โดยกำหนดให้ $E_{\exp ected} = \frac{E_{\max} + E_{\min}}{2}$
5. นำค่า $E_{\exp ected}$ ไปหาฟังก์ชันคลื่น ถ้าไม่สอดคล้องกับเงื่อนไขขอบเขต และจำนวนจุดตัดแกน x ยังไม่เท่ากับ n – 1 ให้กลับไปทำข้อ 4 ใหม่ จนกว่าจะได้ฟังก์ชันคลื่น และพลังงานที่สอดคล้องกับเงื่อนไขขอบเขตและ จำนวนจุดตัดแกน x ของฟังก์ชันคลื่น = n – 1
6. ถ้าพบว่า $\psi(x)$ สอดคล้องกับเงื่อนไขขอบเขต และจำนวนจุดตัดแกน x ที่ได้ มีค่าเท่ากับ n – 1 แสดงว่า $\psi(x)$ เป็นฟังก์ชันเจาะจง และค่า E ที่ได้ในครั้งหลังสุดนี้เป็นค่าเจาะจงที่สอดคล้องกับฟังก์ชันคลื่นนี้เช่นกัน ให้จบการทำงานที่ขั้นตอนนี้

Flowchart 2: แสดงการคำนวณพลังงานของ
อนุภาคเมื่อกำหนดเลขควอนตัม n



นำขั้นตอนทั้งหมด เขียนเป็น source code ภาษาจาวา เพื่อหาคำตอบของสมการชเรอดิงเงอร์ แบบ 1 มิติ ดังนี้

โปรแกรม 4.1 Schrodinger1D.java

```
1:  /**
2:  * Title: Schrodinger1D
3:  * Description: compute energy, eigen state and Wave function
4:  *
5:  * @author: Wach R
6:  *@ version 0.1
7:  * Date : April 23, 2011
8:  * Last updated: May 2, 2011
9:  *============================
10: * Schrodinger Eq :   d^2psi/dx^2 = 2*m/(hbar)^2(E-V)*psi
11: *  define  k^2 = 2*m/(hbar)^2(E-V)
12: * in atomic unit:  hbar = 1, m = 1
13: */
14:  package classes.SE1D;
15:
16:  import static classes.MathTools.Common.CommonConstants.*;
17:  import classes.MathTools.Common.Function;
18:
19: public class  Schrodinger1D
20: {
21:
22:  public static final int NO_ERROR = 0;
23:  public static final int MAXIMUM_NODE=20;
24:  int errorCode = 0;
25:
26: double energy;
27: double potentialMin;
28: double potentialMax;
29: double[] psi;  // wave function of the particle.
30: double[] x;  // positions on x axis
31: double xmin, xmax;  // min and max values of x
32:
33: double tolerance = 1.0e-3;
34: //  double tolerance =DEFAULT_TOLERANCE;
35:  double maxAmplitude;  // maximum amplitude of psi
36:   int firstPoint, lastPoint;
37:   double dx;
38:   Function potential;
39:
40:   double psiState ;   // keep current psi
41:   double  slopeState;  // keep d psi / dx
42:   double  xState;    // keep x value
```

```
43:
44:     public Schrodinger1D (Function potential, double xmin,
                                double xmax, int numberOfPoints) {
45:     this.potential = potential;
46:     psi = new double[numberOfPoints];
47:     x= new double[numberOfPoints];
48:     this.xmin = xmin;
49:     this. xmax = xmax;
50:     dx = (xmax –xmin)/(numberOfPoints –1);
51:     firstPoint = 0;
52:     lastPoint = numberOfPoints;
53:     findMinAndMaxPotential();
54:     }
55:
56:     private void  findMinAndMaxPotential() {
57:                 double xValue = xmin;
58:                 double newPotential;
59:        potentialMin = potential.Of(xValue);
60:                 potentialMax = potentialMin;
61:        for(int i = 0; i <  psi.length; i++) {
62:          x[i] = xValue;
63:          newPotential = potential.Of(xValue);
64:          if(potentialMin > newPotential)
                                potentialMin = newPotential;
65:                  if(potentialMax < newPotential)
                                potentialMax = newPotential;
66:          xValue += dx;
67:        }
68:
69:     }
70:     /**
71:        * finds  the area where the wave function is non–zero.
72:        * and  bounded by the firstPoint and lastPoint.
73:        *
74:        * @param energy
75:        */
76:       protected void findFirstAndLastPoint(double energy) {
77:         double  current_x = xmin;
78:                 double startX=0, stopX=0;
79:
80:     if(( energy <= potentialMin) || (energy >= potentialMax))  {
81:                                 System.out.println( "It's  not a bound state,
                                    check your Potential and Energy again");
82:                                 return;
83:                 }
84:
```

```
85:             // find the first point where E>V
86:
87:             for(int i = 0; i< psi.length;  i++) {
88:                                     if((energy – potential.Of(current_x))>0)  {
89:                 firstPoint = i;
90:                                     startX = current_x;
91:                 break;
92:               }
93:              current_x += dx;
94:             }
95:             current_x = xmax;
96:
97:             // find the last point where E>V
98:             for(int i = psi.length; i>firstPoint; i––) {
99:                                      if((energy – potential.Of(current_x)) > 0) {
100:                                    lastPoint = i;
101:                                    stopX= current_x;
102:                break;
103:              }
104:              current_x –= dx;
105:            }
106:
107:        // Actually, wave function value is not zero immediately
108:        // it still decay at the both sides
109:        // Left side :
110:            double decay = 1;
111:            current_x = xmin+firstPoint*dx;
112:            while(decay > tolerance && firstPoint > 0) {
113:              firstPoint––;
114:             current_x –= dx;
115:                       // this region E < V  the result shoud be minus sign
116:           double k = Math.sqrt(2*Math.abs(energy–potential.Of(current_x)));
117:              decay *= Math.exp(–k*dx);
118:        //System.out.print(" left side: k =  " + k + " decay = " +decay);
119:
120:            }
121:                      // Right side:
122:           decay = 1;
123:           current_x = xmin+lastPoint*dx;
124:            // assume exponential decay on right hand side
125:            while(decay > tolerance && lastPoint < psi.length) {
126:              lastPoint++;
127:              current_x += dx;
128: double k = Math.sqrt(2*Math.abs(energy–potential.Of(current_x)));
129:
130:             decay *= Math.exp(–k*dx);
```

```
131:
132:            }
133:          }
134:
135:          /**
136:           * Solves the Schroedinger Equation with the given energy.
137:           * @param energy double the energy
138:           * @return int node  ( number of zero crossing of wave function
139:           */
140:         public int solveWaveFunction(double energy) {
141:            int node = 0;  // count the number of zero crossings.
142:            boolean slopeChanged = false;
143:            maxAmplitude = 0;
144:            double integral = 0;
145:                        double lastPsiState, lastSlopeState,lastXState;
146:                        double k2;   //  k^2 = 2*(E-V)
147:                        double  k2LastState, k2State;
148:                        double slope;
149:
150:                        this.energy = energy;
151:            findFirstAndLastPoint(energy);
152:            if(lastPoint - firstPoint <3) {
153:         //if number of point in range < 3  cannot use with Numerov method
154:              return 0;
155:            }
156:           // set wave function to zero outside the bound region
157:            for(int i = 0;i<firstPoint;i++) {
158:              psi[i] = 0;
159:              x[i] = xmin+i*dx;
160:            }
161:                        // state of the firstPoint
162:                        lastPsiState=0;
163:                        lastSlopeState = 0;
164:                        lastXState =xmin+firstPoint*dx;
165:                        psi[firstPoint]=lastPsiState;
166:                        x[firstPoint] = lastXState;
167:                        k2LastState = 2.0*(energy - potential.Of(lastXState));
168:                        // state of the firstPoint +1
169:                        psiState = dx;
170:           slopeState = 1.0;  // initial derivative of psi by dx
171:           xState =  xmin + (firstPoint+1)*dx;   // initial value of x
172:                        psi[firstPoint+1]=psiState;
173:                        x[firstPoint+1] = xState;
174:                        k2State = 2.0*(energy - potential.Of(xState));
175:                        slopeState = (psiState - lastPsiState)/dx;
176:         // find wave function (psi) with Numberov method
```

```
177:    // the differential eq must be in a form
178:    //                                    d^2 psi /dx^2  = –k^2 (psi)
179:    //                                     define  k^2 = 2*m/(hbar)^2(E–V)
180:    // ------------------------------------
181:    double  c =  dx*dx/12.0;  // coeficient for Numerov Algorithm
182:        for(int i = firstPoint+2; i<lastPoint; i++) {
183:                                x[i]= xmin+i*dx;
184:                                k2 = 2.0*(energy – potential.Of(x[i]));
185:    // System.out.print ( " i =  "  + i + "      k in loop =  " + k2);
186:
187:                                psi[i] = ((2.0 –10.0*c*k2State)*psiState –
                                    (1+ c*k2LastState)*lastPsiState)/(1+c*k2);
188:                                slope = (psi[i] – psiState)/dx;
189:
190:                                // keep previous values for next step
191:                    lastPsiState=psiState;
192:                    lastSlopeState = slopeState;
193:                    lastXState =xState;
194:                    k2LastState = k2State;
195:
196:     // Now store current values
197:                    psiState = psi[i];
198:                    k2State = k2;
199:                     xState = x[i] ;
200:                     slopeState = slope;
201:
202:            if(maxAmplitude<Math.abs(psiState)) {
203:              maxAmplitude = Math.abs(psiState);
204:            }
205:    // Normalize  wave function  with trapezoidal rule
206:    //  Integrate f(x) dx =  deltaX ( summation fi – f0/2 –fn/2)
207:
208:                     integral += psiState*psiState;
209:
210:                    if(slopeState*lastSlopeState < 0)
211:              slopeChanged = true;
212:
213:                     if(psiState*lastPsiState < 0)  {
214:              node++;
215:              slopeChanged = false;
216:            }
217:            if(Math.abs(psiState)>1.0e12) { //prevent diverging so much
218:              for(int m = i+1;m<lastPoint; m++)
219:                psi[m] = psiState;
220:                                x[m] = xmin + m*dx;
221:              }
```

```
222:                 break;
223:               }
224:             } // end of ----->  for(int i = firstPoint+2
225:
226:             integral *= dx;
227:                       integral = integral –( psi[firstPoint]/2.0 +
                          psi[lastPoint–1]/2.0)*dx;
228:                       // find the magnitude of wave function
229:                       double sqrtIntegral = Math.sqrt(integral);
230:
231:                       // fill the rest point at the right side
232:             for(int i = lastPoint; i < psi.length; i++) {
233:               psi[i] = psi[lastPoint–1];
234:               x[i] = xmin+i*dx;
235:             }
236:          /*
237:             // check to see if last value is close enough to a crossing
238:if(slopeChanged&&(Math.abs(phi[phi.length–1])<=tolerance*maxAmp)) {
239:               crossing++;
240:             }
241:          */
242:
243:             normalize(sqrtIntegral);
244:
245:                       // check normalize it shoulld equals  1
246:          // double sumOfPsi2 =0;
247:          //             for(int i = firstPoint; i <lastPoint; i++) {
248:          //               sumOfPsi2  += psi[i]*psi[i];
249:          //             }
250:          //             System.out.println(" sum of  Psi ^2 = " + sumOfPsi2*dx );
251:             return node;
252:           }
253:          //  Normalize wave function
254:           private void normalize(double magnitude) {
255:             if(magnitude==0) {
256:               return;
257:             }
258:             for(int i = 0; i< psi.length; i++) {
259:               psi[i] /= magnitude;
260:               psiState /= magnitude;
261:              slopeState /= magnitude;
262:             }
263:             maxAmplitude /= magnitude;
264:           }
265:
266:           /**
```

```
267:        * Solves the eigen energy  with the given energy level
268:        * @param int quantumNumber
269:        * @return int node  ( number of zero crossing of wave function
270:        */
271:      public double[ ] solveEigenEnergy( int quantumNumber)
272:      {
273:      double eMin;
274:      double eMax;
275:      int nodeCount=0;
276:      int counter = 0;
277:
278:                  if(quantumNumber > MAXIMUM_NODE)  {
279:                  System.out.println("Quantum  Number must not  greater
                     than  " + MAXIMUM_NODE);
280:                  return new double[psi.length];
281:      }
282:                  eMin = potentialMin;
283:                  eMax = eMin +1;
284:      while (counter < 20 && nodeCount <= quantumNumber) {
285:                  nodeCount = solveWaveFunction(eMax);
286:
287:      //Increase energy until the number of node is greater
         than quantum number
288:                  eMax  = eMax +( eMax-eMin);
289:                  counter++;
290:      }
291:      counter = 0;
292:
293:      do{
294:                  double tempEnergy = (eMax + eMin)/2.0;
295:                  int crossing = solveWaveFunction(tempEnergy);
296:                    System.out.println( counter + "  energy =  " +
                     tempEnergy + " and crossing zero =    " + crossing);
297:
298:                  if((crossing == quantumNumber) &&
                     (Math.abs(psi[psi.length-1]) <= tolerance*maxAmplitude)) {
299:                              errorCode = 0;
300:      //System.out.println("in IF #1:  psi[psi.lenth-1] = " +
                     psi[psi.length-1] + "      tolerance*maxAmp = "
                     + tolerance*maxAmplitude );
301:                              return psi;
302:                  }
303:      if((crossing > quantumNumber) || ((crossing == quantumNumber)
         &&(checkEvenNumber(crossing)*psi[psi.length-1] > 0)) ) {
304:                              eMax = tempEnergy;
305:
```

```
306:
307:                    }
308:                    else {
309:                                    eMin = tempEnergy;
310:
311:
312:                    }
313:                    counter ++;
314:                    System.out.println();
315:        } while (counter < 32 && (eMax–eMin )> ZERO_APPROACH);
316:        errorCode = 1;
317:        return new double[psi.length];
318:        }
319:          /**
320:           * Gets the x coordinates
321:           */
322:          public double[] getXCoordinates() {
323:            return x;
324:          }
325:          public double[] getWaveFunction() {
326:            return psi;
327:          }
328:        public double getEnergy() {
329:                    return energy;
330:        }
331:         /**
332:           * Gets the error code from computation.
333:           * @return int
334:           */
335:          public int getError() {
336:            return errorCode;
337:          }
338:          private int checkEvenNumber(int number) {
339:                      if(number%2==0)
340:                                    return 1;  // Even
341:                      else
342:                                    return –1; //Odd
343:          }
344:        }
345:
```

โปรแกรม 4.1 Schrodinger1D.java มีส่วนที่เป็นการฟังก์ชันคลื่น (ใน flowchart ที่ 1) และส่วนที่หาขนาดพลังงานที่เลขควอนตัมใดๆ (ใน flowchart ที่ 2)

บรรทัดที่ 140 ถึง 264 เป็น method solveWaveFunction มีรูปแบบการใช้งานดังนี้

```
public int solveWaveFunction(double energy)
```

เมธอดนี้ใช้หาฟังก์ชันคลื่นของอนุภาคที่มีพลังงาน เท่ากับค่า energy ลำดับการทำงานของเมธอดนี้เป็นไปตามที่เขียนไว้ใน Flowchart ที่ 1 มีรายละเอียดปลีกย่อยได้แก่ บรรทัดที่ 180 – 224 เป็นการหาค่า ψ_i โดยใช้วิธีหาผลเฉลยของสมการอนุพันธ์อันดับสองของ นูเมอรอฟ (Numerov) บรรทัดที่ 213–215 เป็นการตรวจสอบว่า ψ_i ตัดแกน x หรือไม่ ถ้ามีการตัดแกน x จะนับจำนวนครั้งที่ตัดแกน x ไว้ในตัวแปรที่ชื่อว่า node บรรทัดที่ 217 –223 เป็นตรวจสอบการลู่ออกของฟังก์ชันคลื่น ψ_i ถ้าค่า ψ_i ที่คำนวณได้มีค่ามากกว่า 10^{12} การวนรอบหาฟังก์ชันคลื่นจะหยุดคำนวณทันที บรรทัดที่ 205 –208 เป็นการปริพันธ์ของฟังก์ชันคลื่น เพื่อเตรียมไว้สำหรับ normalize โดยอาศัยวิธีรวมพื้นที่สี่เหลี่ยมคางหมูเล็ก ๆ ทั้งหมด จากนั้นนำผลลัพธ์ที่ได้จากการหาปริพันธ์ไปทำการ normalize โดยใช้เมธอด normalize ในบรรทัดที่ 254 –264

เมธอด solveWaveFunction ส่งค่าคืนกลับเป็นจำนวนจุดตัดของฟังก์ชันคลื่นที่ผ่านแกน x

บรรทัดที่ 271 ถึง 318 เป็น method solveEigenEnergy มีรูปแบบการใช้งานดังนี้

```
public double[ ] solveEigenEnergy( int quantumNumber)
```

เมธอดนี้ใช้หาขนาดของพลังงานที่ระดับพลังงานหรือเลขควอนตัม n ใด ๆ ลำดับการทำงานเป็นไปตาม Flowchart 2 มีการเรียกใช้ method solveWaveFunction ในบรรทัดที่ 295 การวนรอบหาฟังก์ชันคลื่นจะดำเนินไปเรื่อย ๆ จนกว่าจะได้ฟังก์ชันคลื่นที่สอดคล้องเงื่อนไขขอบเขตและสอดคล้องกับเลขควอนตัม n ขนาดของพลังงานที่ได้จะเป็นค่าเจาะจง เมธอดนี้ส่งค่าคืนกลับเป็นเซตของฟังก์ชันคลื่นที่สอดคล้องกับเงื่อนไขขอบเขตและเป็นฟังก์ชันเจาะจง

ในการทดสอบการทำงานของคลาส Schrodinger1D ทำได้โดยสร้างโปรแกรมเรียกใช้งานคลาสดังกล่าว ตัวอย่างต่อไปนี้เป็นการสร้าง Application ชื่อ NSEApp.java (Appllication for Schrodinger equation with quantum number n) ใช้พลังงานศักย์เป็น Harmonic oscillator เป็นตัวอย่าง รายละเอียดของโปรแกรมมีดังนี้

โปรแกรม 4.2 NSEApp.java

```
1:  /**
2:  * Title: NSEApp.java  (Schrodinger Application)
3:  * Description : test for Schrodinger App
4:  * @author: Wach R
5:  *@ version 0.1
```

```
 6:      * Date : April 28, 2011
 7:      *============================
 8:       */
 9:       package  JApp.SE1D;
10:
11:      import java.util.*;
12:      import java.io.*;
13:      import classes.SE1D.*;
14:
15:      public class NSEApp {
16:
17:        /**
18:          * Starts the Java application.
19:          * @param args[]  the input parameters
20:          */
21:        public static void main(String[] args) throws Exception {
22:          int numberOfPoints = 300;
23:          double xmin = -5, xmax = +5;
24:                    int n; // quantum number
25:                    double en;
26:                    String fileName = "NPsi";
27:                    int crossing=0;
28:
29:      System.out.println("\nThis program finding eigen energy of
         a particle\n in harmonic oscillator");
30:      System.out.print(" You must input n (quantum no. (from 1 to 30))
         ---------->");
31:                                Scanner  console = new Scanner(System.in);
32:                                          n = console.nextInt();
33:
34:           Schrodinger1D  se = new Schrodinger1D(new Quadratic(),
                    xmin, xmax, numberOfPoints);
35:      //   Schrodinger1D  se = new Schrodinger1D(new SquareWell(2.0,
                    -10.0), xmin, xmax, numberOfPoints);
36:
37:                    double[] psi = se.solveEigenEnergy(n);
38:          double[] x = se.getXCoordinates();
39:
40:                    if (se.getError()==se.NO_ERROR) {
41:             System.out.println("energy="+se.getEnergy());
42:                                PrintWriter w = new PrintWriter (new
                                          FileOutputStream(fileName+n), true);
43:                                                    for(int i = 0; i < x.length ; i++)
44:                                                              w.println( x[i]+"\t"+psi[i]);
45:
46:             } else {
```

```
47:                 System.out.println ("There must be some errors");
48:               }
49:         }
50:         }
```

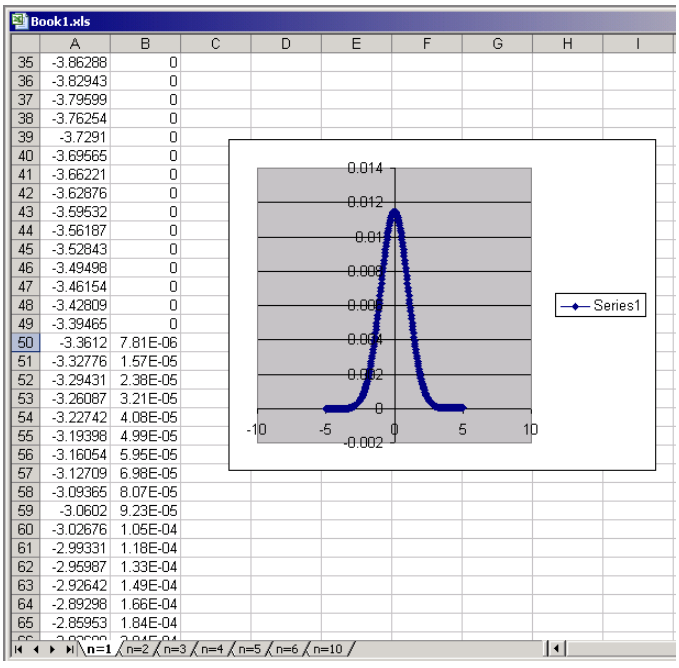
โปรแกรม 4.2 นี้ แสดงตัวอย่างการจำลองอนุภาคในบ่อศักย์กรณีที่เป็น Harmonic oscillator การทำงานของโปรแกรม NSEApp.java เริ่มต้นด้วยการป้อนค่า n ที่ต้องการทราบพลังงาน

บรรทัดที่ 34 คือบรรทัดที่กำหนดแบบของบ่อศักย์ V(x) ถ้าต้องการทดสอบกับบ่อศักย์ชนิดอื่น ๆ ให้เปลี่ยน ฟังก์ชันของบ่อศักย์ที่บรรทัดนี้

บรรทัดที่ 37 โปรแกรมจะนำค่า n ไปคำนวณพลังงาน โดยเรียกใช้ฟังก์ชัน se.solveEigenEnergy(n) แสดงผลพลังงานที่คำนวณได้ในบรรทัดที่ 41

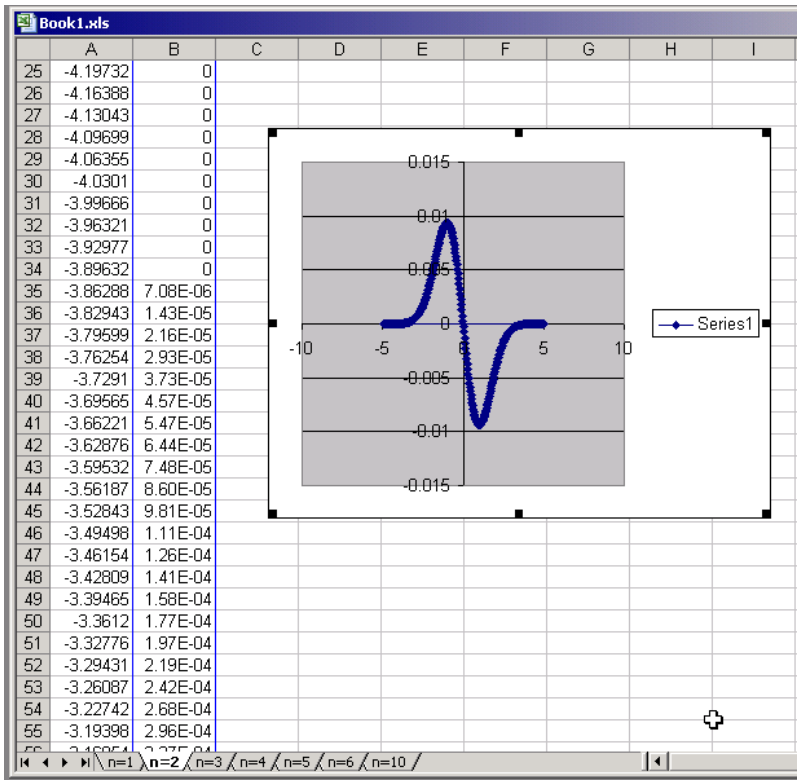
บรรทัดที่ 42 – 45 เป็นการบันทึกค่าฟังก์ชันคลื่นที่คำนวณได้ลงในแฟ้มข้อมูล ชื่อ NPsi1, NPsi2, ... ชื่อแฟ้มข้อมูลจะเปลี่ยนไปตามเลขควอนตัมที่ป้อนทางอินพุต การบันทึกเก็บไว้ในไฟล์จะเป็นประโยชน์สำหรับการนำไปตรวจสอบฟังก์ชันคลื่นว่ามีคุณสมบัติสอดคล้องกับเงื่อนไขขอบเขตหรือไม่ สามารถนำไปพล็อตกราฟ โดยใช้โปรแกรมตารางคำนวณ Excel เพื่อศึกษารูปกราฟว่าเป็นไปตามทฤษฎีหรือไม่

จากรูปที่ 4.1 และ 4.2 เป็นการนำฟังก์ชันคลื่นที่คำนวณได้ กรณี n = 1 และ n = 2 ไปพล็อตกราฟในโปรแกรม Excel จะเห็นว่าจะได้เส้นกราฟสอดคล้องกับทฤษฎี กล่าวคือ ฟังก์ชันคลื่นจะหายไปตรงผนังของบ่อ จำนวนจุดตัดแกน x จะมีค่าเท่ากับ n–1

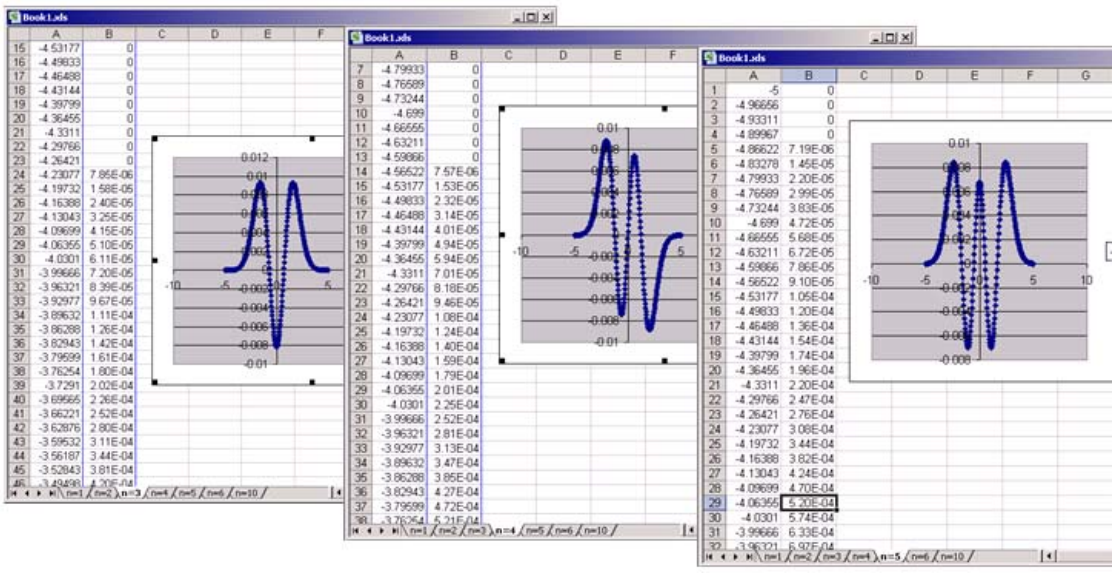


	A	B
35	-3.86288	0
36	-3.82943	0
37	-3.79599	0
38	-3.76254	0
39	-3.7291	0
40	-3.69565	0
41	-3.66221	0
42	-3.62876	0
43	-3.59532	0
44	-3.56187	0
45	-3.52843	0
46	-3.49498	0
47	-3.46154	0
48	-3.42809	0
49	-3.39465	0
50	-3.3612	7.81E-06
51	-3.32776	1.57E-05
52	-3.29431	2.38E-05
53	-3.26087	3.21E-05
54	-3.22742	4.08E-05
55	-3.19398	4.99E-05
56	-3.16054	5.95E-05
57	-3.12709	6.98E-05
58	-3.09365	8.07E-05
59	-3.0602	9.23E-05
60	-3.02676	1.05E-04
61	-2.99331	1.18E-04
62	-2.95987	1.33E-04
63	-2.92642	1.49E-04
64	-2.89298	1.66E-04
65	-2.85953	1.84E-04

รูปที่ 4.1 แสดงการนำฟังก์ชันคลื่นกรณี n = 1 พล็อตกราฟในโปรแกรม Excel



รูปที่ 4.2 แสดงการนำฟังก์ชันคลื่นกรณี n = 2 พล็อตกราฟในโปรแกรม Excel



รูปที่ 4.3 กราฟของฟังก์ชันคลื่น กรณี n =3, 4 และ 5

ผลลัพธ์จากการจำลองอนุภาคในบ่อศักย์แบบสี่เหลี่ยม (1 dimensional finite rectangular well)

อนุภาคมวล m เคลื่อนที่ในบ่อศักย์ที่มีความกว้างเท่ากับ 2a ความลึกของบ่อศักย์ V ดังนี้

$$V = \begin{cases} -V_0 & if \; -a < x < a \\ 0 & if \quad |x| \geq a \end{cases}$$

นำไปเขียนเป็นโปรแกรม เพื่อเรียกใช้ฟังก์ชันของบ่อศักย์ดังนี้

โปรแกรม 4.3 SquareWell.java

```
1:  /**
2:  * Title: SquareWell.java
3:  * Description Square well potential.
4:  *
5:  * @author: Wach R
6:  *@ version 0.1
7:  * Date : April 24, 2011
8:  *============================
9:   */
10:   package classes.SE1D;
11:
12:  import classes.MathTools.Common.Function;
13:
14:  public class SquareWell  extends Function {
15:  double halfWidth;  // halfWidth of potential well( –a <= x <= a )
16:                  double dept;//the depth of well  must be minus sign
17:  public          SquareWell( double width, double dept) {
18:                                  halfWidth = width;
19:                                  this.dept = dept;
20:                  }
21:
22:    public double Of(double x) {
23:                    if(  Math.abs(x) >= halfWidth)
24:                                    return 0;
25:                    else
26:                                                    return  dept;
27:    }
28:
29:  }
30:
```

การคำนวณพลังงาน (E) ของอนุภาคในบ่อศักย์ที่ระดับพลังงาน n ใด ๆ โดยวิธีวิเคราะห์สามารถหาได้จาก สมการ (2.32)

$$2a\sqrt{\frac{2m}{\bar{h}^2}(V_0 - E)} = (n-1)\pi + 2\tan^{-1}\sqrt{\frac{E}{V_0 - E}}$$

ในหน่วยอะตอม สมการจะถูกลดรูปเหลือเพียง

$$2a\sqrt{2(V_0 - E)} = (n-1)\pi + 2\tan^{-1}\sqrt{\frac{E}{V_0 - E}} \qquad (4.1)$$

ในที่นี้จะใช้สมการ (4.1) ในการคำนวณหาพลังงาน E เมื่อกำหนดค่า n =1, 2, 3 ..., 6 ให้ความกว้างของบ่อ เท่ากับ 4 หน่วย (a = 2) และความลึกของบ่อศักย์เท่ากับ 10 หน่วย เนื่องจากสมการ (4.1) เป็นสมการประเภท transcendental function ไม่สามารถใช้วิธีวิเคราะห์ใด ๆ คำนวณหารากสมการ (ในที่นี้คือ E) ได้ จะต้องใช้วิธีคำนวณเชิงตัวเลขเข้ามาช่วย ซอฟต์แวร์ที่จะใช้คำนวณในงานวิจัยนี้คือ MathematicA รุ่น 5.1 และใช้การหารากสมการโดยวิธีแบ่งครึ่งช่วง (Bisection method) โดยเขียนเป็นโปรแกรมคอมพิวเตอร์แล้วป้อนข้อมูลที่กำหนดให้ เพื่อใช้ในการคำนวณ (ดูรายละเอียดการคำนวณของโปรแกรม MathematicA และโปรแกรมต้นฉบับ (source code) ที่ใช้วิธีแบ่งครึ่งช่วงหารากสมการ ในภาคผนวก 3) ผลที่ได้ปรากฏดังในตารางที่ 4.1 ดังนี้

n (ระดับพลังงาน)	E จาก MathematicA	E จากวิธีแบ่งครึ่งช่วง	E จากแบบจำลอง ฯ
1	–9.7507	–9.750697	–9.752279
2	–9.00542	–9.005423	–9.011693
3	–7.77297	–7.772970	–7.786852
4	–6.07186	–6.071860	–6.095848
5	–3.94146	–3.941455	–3.977063
6	–1.49125	–1.491250	–1.536339

ตาราง 4.1 เปรียบเทียบพลังงานของอนุภาค (E) ที่ n ค่าต่างๆ กัน ของบ่อศักย์แบบสี่เหลี่ยม

ผลลัพธ์ที่ได้จากการจำลองอนุภาคในบ่อศักย์แบบสี่เหลี่ยมที่มีความลึก –10 หน่วย และความกว้าง 4 หน่วย จะได้พลังงานที่เป็นค่าเจาะจง ที่ n = 1, 2, ..., 5, 6 มีค่าใกล้เคียงกับค่าที่ได้จากการคำนวณโดยใช้โปรแกรมสำเร็จรูป MathematicA และวิธีคำนวณเชิงตัวเลขโดยวิธีแบ่งครึ่งช่วง ถูกต้องที่ทศนิยมตำแหน่งที่ 1

ผลลัพธ์จากการจำลองอนุภาคในบ่อศักย์เป็นฮาร์มอนิก ออสซิลเลเตอร์

(1 dimensional harmonic oscillator)

อนุภาคเคลื่อนที่ในบ่อพลังงานศักย์แบบฮาร์มอนิก ออสซิลเลเตอร์ ซึ่งมีรูปสมการดังนี้

$$V(x) = \frac{1}{2} m\omega^2 x^2$$

ω คือความเร็วเชิงมุมของการสั่นของอนุภาคในหน่วย radian/sec

นำไปเขียนเป็นโปรแกรม เพื่อเรียกใช้ฟังก์ชันของบ่อศักย์ดังนี้

โปรแกรม 4.4 Quadratic.java

```
1:  /**
2:  * Title: Quadratic.java
3:  * Description Simple harmonic oscillator potential.
4:  *
5:  * @author: Wach R
6:  *@ version 0.1
7:  * Date : April 23, 2011
8:  *============================
9:   */
10:  package classes.SE1D;
11:
12:  import classes.MathTools.Common.Function;
13:
14: public class Quadratic  extends Function {
15:    public double Of(double x) {
16:    return (x*x)/2;
17:    }
18:    public  double DerivativeOf(double x) {  return 0.0; }
19:
20:
21: }
22:
```

พลังงานของอนุภาคในบ่อศักย์ มีค่าไม่ต่อเนื่อง เมื่อ n = 1, 2, 3, ... หาได้จาก

$$E_{n-1} = \bar{h}\omega(n - \frac{1}{2}) \qquad (4.2)$$

ผลลัพธ์ที่ได้จากการจำลองอนุภาคในบ่อศักย์แบบฮาร์มอนิก ออสซิลเลเตอร์ ที่กำหนดให้ $\bar{h} = \omega = 1$ จะได้พลังงานที่เป็นค่าเจาะจง ที่ n = 1, 2, ..., 5, 6 เปรียบเทียบกับค่าที่ได้จากการคำนวณตามสมการ (4.2) ดังตารางต่อไปนี้ พบว่าผลลัพธ์ที่ได้จากแบบจำลองมีค่าเท่ากับผลที่ได้จากการคำนวณโดยวิธีวิเคราะห์ทุกประการ

n (ระดับพลังงาน)	E จากวิธีวิเคราะห์	E จากแบบจำลอง ฯ
1	0.50	0.50
2	1.50	1.50
3	2.50	2.50
4	3.50	3.50
5	4.50	4.50
6	5.50	5.50

ตาราง 4.2 พลังงานของอนุภาค (E) ที่ n ค่าต่าง ๆ กันของบ่อศักย์แบบฮาร์มอนิก ออสซิลเลเตอร์

ผลลัพธ์จากการจำลองอนุภาคในบ่อศักย์แบบอื่น ๆ

ผลการคำนวณพลังงานของอนุภาคที่สถานะ n (ตั้งแต่ 1 ถึง 10) ของบ่อศักย์แต่ละแบบ โดยกำหนดค่าคงที่ของพลังงานศักย์ เพื่อให้การแสดงผลทางกราฟิกอยู่ในขอบเขตของจอภาพ ดังแสดงไว้ในตาราง ผลลัพธ์ในการคำนวณที่ได้จากโปรแกรม สามารถเขียนเป็นตารางได้ดังนี้

สถานะ n	Square well	Parabolic	Triangular	Square tangent	Square hyperbolic cosine
	$V=0 \quad \{\lvert x\rvert>2$ $V=-10 \quad \{-2\le x\le 2$	$V=\frac{1}{2}x^2$	$V=2.0\lvert x\rvert$	$V=\frac{1}{560}\tan^2 x$ $-\frac{\pi}{2}\le x\le\frac{\pi}{2}$	$V=\frac{-10}{\cosh^2 x}$
1	−9.752 280	+0.500 001	+1.283 745	+0.397 526	−7.999 999
2	−9.011 696	+1.500 001	+2.945 831	+1.578 424	−4.999 999
3	−7.786 852	+2.500 001	+4.092 519	+3.507 880	−2.000 000
4	−6.095 848	+3.500 002	+5.150 50	+6.107 141	−0.498 165
5	−3.977 063	+4.500 014	+6.073 163	+9.112 893	+0.136 219
6	−1.536 339	+5.500 102	+6.957 181	+10.363 892	+0.488 270
7	+0.369 385	+6.500 613	+7.775 388	+10.466 563	+0.997 425
8	+0.567 383	+7.502 947	+8.593 503	+11.360 108	+1.639 515
9	+1.154 297	+8.511 492	+9.418 333	+11.729 493	+2.402 942
10	+2.072 021	+9.536 639	+10.318 601	+12.688 721	+3.281 116

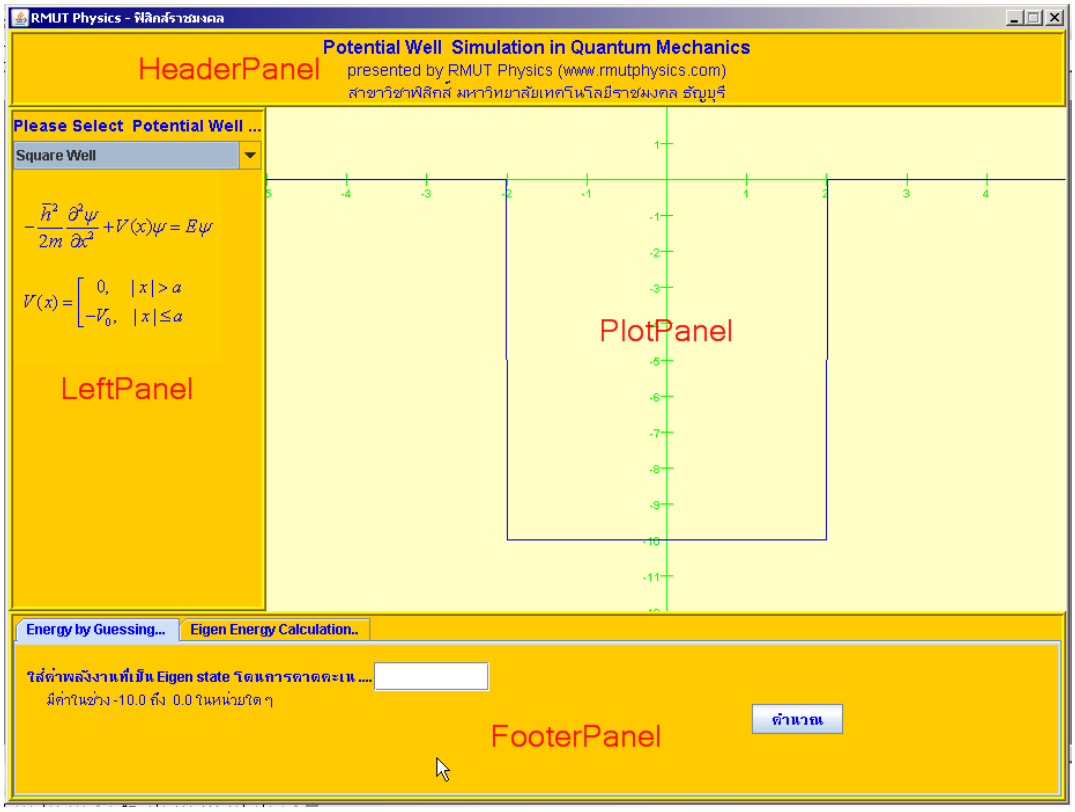
พลังงานที่ n ที่พิมพ์ไว้ด้วยตัวหนานั้น โปรแกรมจะแจ้งเตือนว่าระดับพลังงานที่ n ดังกล่าวไม่เป็น bound state

4.2 ผลลัพธ์การแสดงภาพกราฟิกของบ่อศักย์และฟังก์ชันคลื่นบนจอภาพ

การแสดงภาพกราฟิกในงานวิจัยนี้ ได้ใช้ Java Swing ซึ่งเป็นคลาสที่ติดตั้งพร้อมกับ JDK สามารถเรียกใช้งานคลาสต่าง ๆ เกี่ยวกับกราฟิก ได้ทันทีโดยไม่ต้องเขียนเมธอดขึ้นมาใหม่ แต่อาจต้องมีการ override เมธอดบางเมธอด เพื่อให้ได้ผลลัพธ์ตามจุดมุ่งหมายของงานวิจัย

Panel ต่าง ๆ ที่ใช้แสดงผลทั้งหมดสืบทอดมาจากคลาส JPanel โดยมี BasePanel เป็น panel หลักรองรับ panel อื่น ๆ ทั้งหมด การซ้อนทับของ panel มีลักษณะเป็นชั้น ๆ ประกอบด้วย HeaderPanel จะอยู่ด้านบนสุดใช้แสดงชื่อเรื่องและบ่อศักย์ที่กำลังเลือกศึกษา LeftPanel จะอยู่ด้านซ้ายมือของผู้ใช้ ส่วนนี้จะใช้แสดงตัวเลือกฟังก์ชันของบ่อศักย์ FooterPanel จะอยู่ด้านล่างสุดของจอภาพ เป็นส่วนที่แสดงการรับค่าจากคีย์บอร์ด และใช้เลือกสถานะ n ของพลังงาน PlotPanel จะเป็น panel ที่ใช้แสดงรูปกราฟของบ่อศักย์ และแสดงฟังก์ชันคลื่นที่สอดคล้องกับพลังงานที่ผู้ใช้เลือก

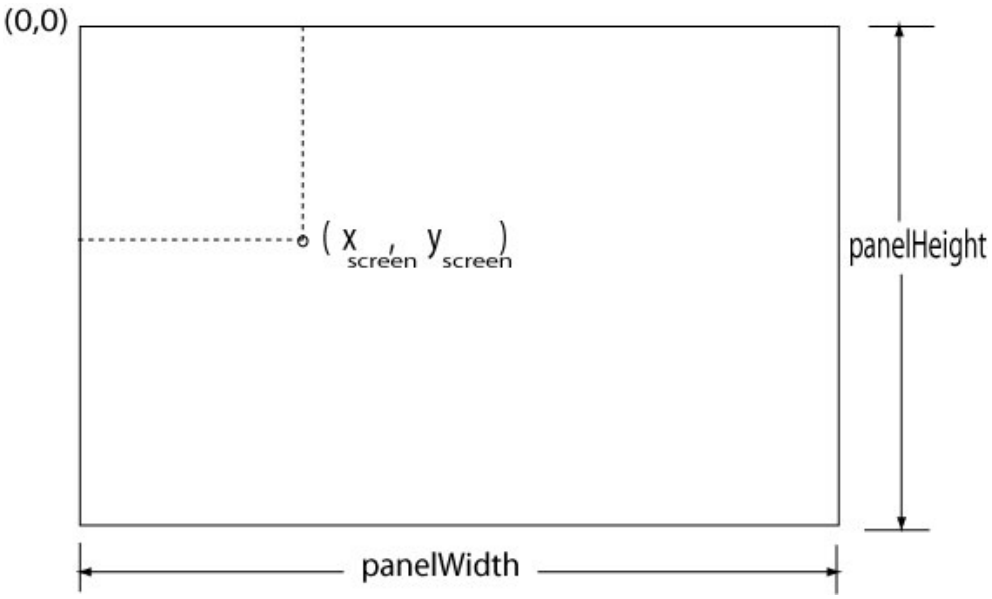
เมื่อแสดงผลบนจอภาพ จะเป็นดังรูปที่ 4.4



รูปที่ 4.4 เมื่อโปรแกรมทำงานจะได้ panel แต่ละ panel วางซ้อนบน BasePanel

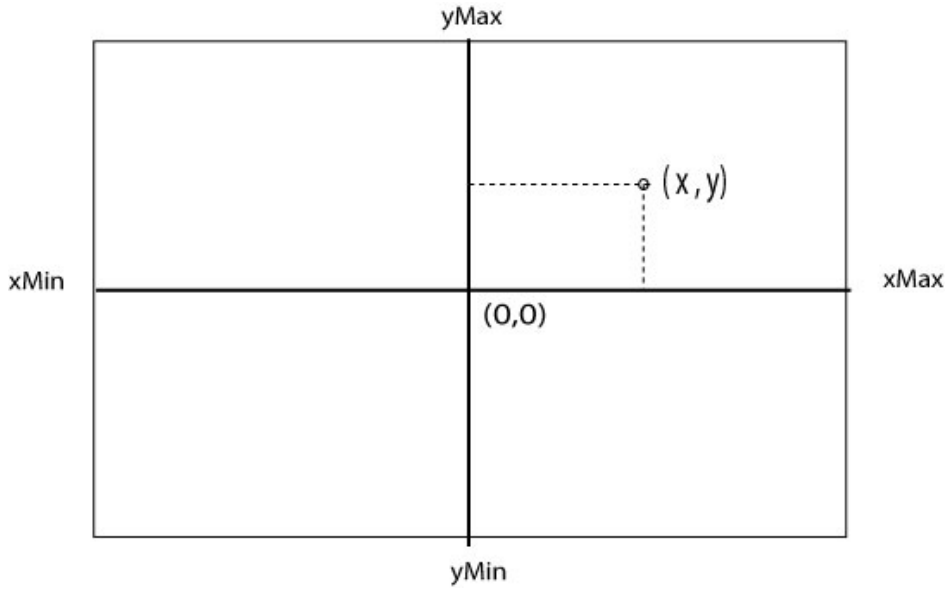
การแสดงภาพกราฟิกของบ่อศักย์และฟังก์ชันคลื่น จะแสดงอยู่บน Plot Panel ขนาดของพื้นที่ของPlotPanel จะเปลี่ยนแปลงไปตามขนาดของจอภาพที่ผู้ใช้ใช้งาน จุดโคออร์ดิเนตของระบบพิกัดฉากที่ใช้ในการแสดงผลบนจอภาพจะแตกต่างจากจุดโคออร์ดิเนตของระบบพิกัดฉากที่ใช้ในการพล็อตกราฟทางคณิตศาสตร์ ในการวาดรูปกราฟิกบน PlotPanel จึงต้องมีการแปลงจุดโคออร์ดิเนตจากผลการคำนวณทางคณิตศาสตร์ให้เป็นจุดโคออร์ดิเนตบนจอภาพเสียก่อนที่จะวาดภาพลงไป

จุดโคออร์ดิเนตของ PlotPanel บนจอภาพจะเริ่มต้น จุด (0,0) ที่มุมบนซ้าย x มีค่าเป็นบวกโดยนับไปทางด้านขวามือ y มีค่าเป็นบวกเมื่อนับลงไปทางด้านล่าง ดังรูป



รูปที่ 4.5 แสดงตำแหน่ง x, y บนจอภาพ

ในการพล็อตกราฟทางคณิตศาสตร์ จะตั้งแกน x และ y ตรงกึ่งกลางจอภาพ ค่า x มีค่าเป็นลบเมื่อนับมาทางด้านซ้าย และมีค่าเป็นบวก เมื่อนับไปทางด้านขวา ส่วนค่า y มีค่าเป็นบวกถ้านับในทิศทางพุ่งขึ้น และมีค่าเป็นลบเมื่อนับในทิศทางพุ่งลง การนำจุดโคออร์ดิเนตที่ได้จากการคำนวณมาพล็อตลงบนพื้นที่ PlotPanel จึงมิสามารถกระทำได้โดยตรง ต้องอาศัยการแปลงดังสมการต่อไปนี้



รูปที่ 4.6 แสดงจุดโคออร์ดิเนตทางคณิตศาสตร์ที่พล็อตลงในกระดาษกราฟ

เมื่อทราบตำแหน่ง (x, y) ใด ๆ สามารถแปลงเป็น ตำแหน่ง (x, y) บนพื้นที่ PlotPanel ของจอภาพ (ต่อไปจะเรียกเป็น x_{screen} และ y_{screen}) ดังนี้

$$x_{screen} = (x + xMin) \times \frac{panelWidth}{(xMax - xMin)}$$

$$y_{screen} = (yMax - y) \times \frac{panelHeight}{(yMax - yMin)}$$

ตำแหน่งที่จะวาดแกน x (xAxisRow) และ แกน y (yAxisCol) หาได้จาก

$$xAxisRow = round(\frac{yMax}{yMax - yMin}) \times panelHeight$$

$$yAxisCol = round(\frac{-xMin}{xMax - xMin}) \times panelWidth$$

การวาดแกน x และแกน y ใช้คำสั่งต่อไปนี้

แกน x: bg.drawLine(0, xAxisRow, panelWidth, xAxisRow);

แกน y: bg.drawLine(yAxisCol,0, yAxisCol, panelHeight);

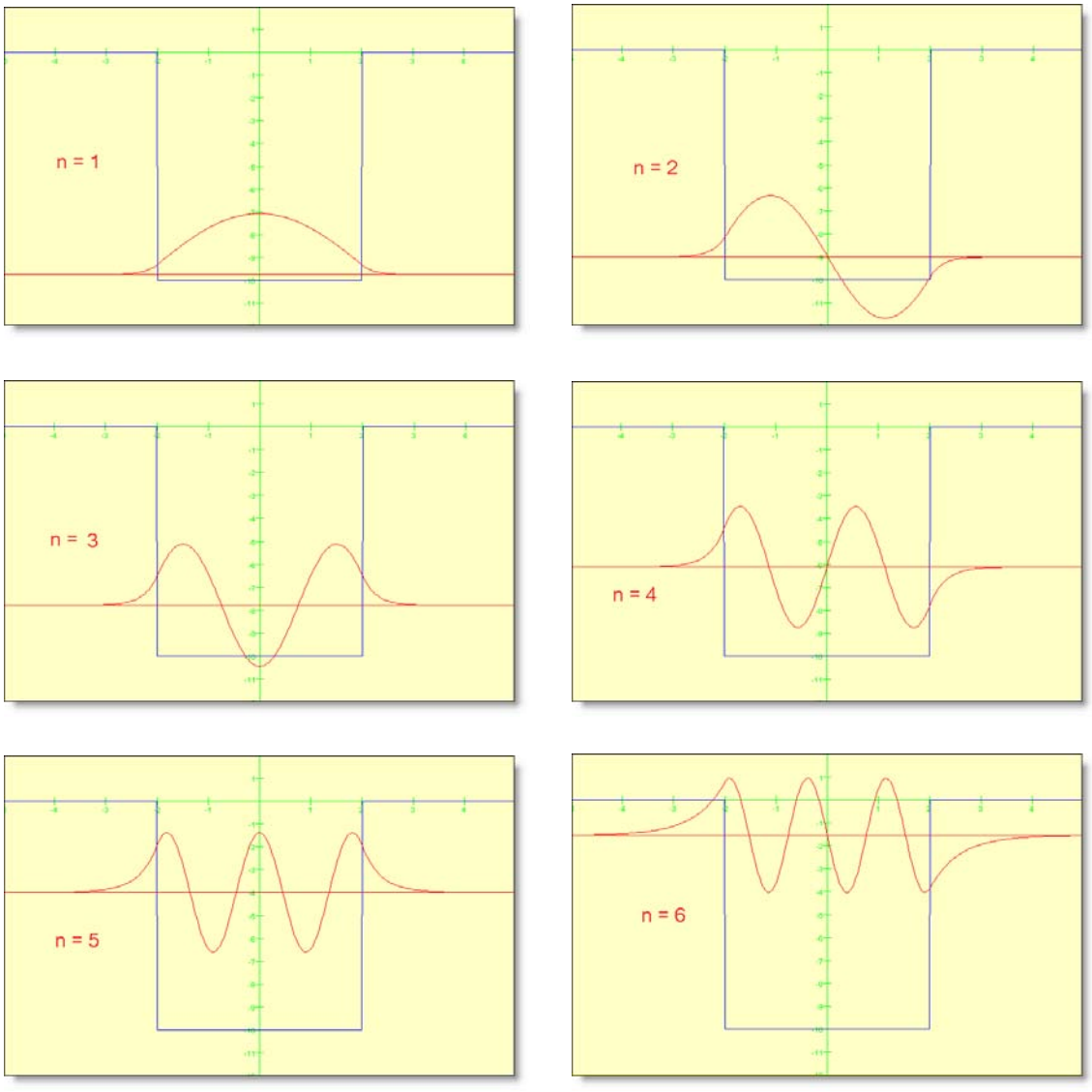
ในกรณีที่จะวาดฟังก์ชันคลื่น ให้แกน x สามารถเลื่อนขึ้นเลื่อนลงได้ตามระดับของพลังงาน สามารถทำได้โดยกำหนดค่า offset ของแกน y ดังนี้

$$y_{offset} = round(\frac{(yMax - y)}{yMax - yMin} \times panelHeight) - xAxisRow$$

รูปกราฟของบ่อศักย์และฟังก์ชันคลื่นของอนุภาคที่สถานะ n ต่าง ๆ โดยเลือกแสดงเพียงบางค่าดังนี้

บ่อศักย์แบบสี่เหลี่ยม มีความกว้าง 4 หน่วย มีความลึก –10 หน่วย เขียนเป็นสมการได้ดังนี้

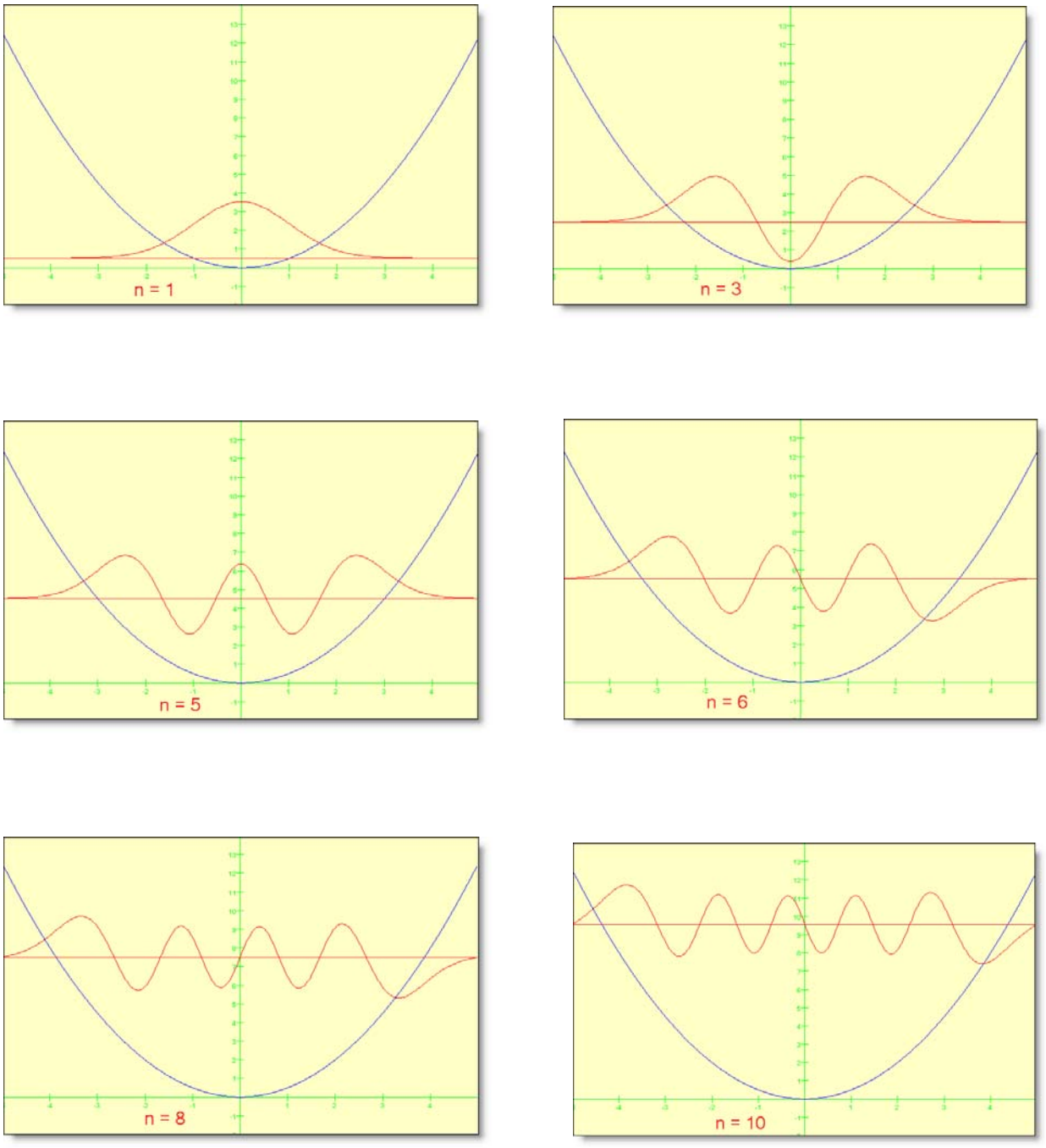
$$\begin{aligned} V &= 0 \quad \{ |x| > 2 \\ V &= -10 \quad \{ -2 \le x \le 2 \end{aligned}$$



รูปที่ 4.7 ฟังก์ชันคลื่นในบ่อศักย์แบบสี่เหลี่ยมที่มีความลึกจำกัด

บ่อศักย์แบบพาราโบลา หรือ ฮาร์มอนิก ออสซิลเลเตอร์ มีรูปสมการดังนี้

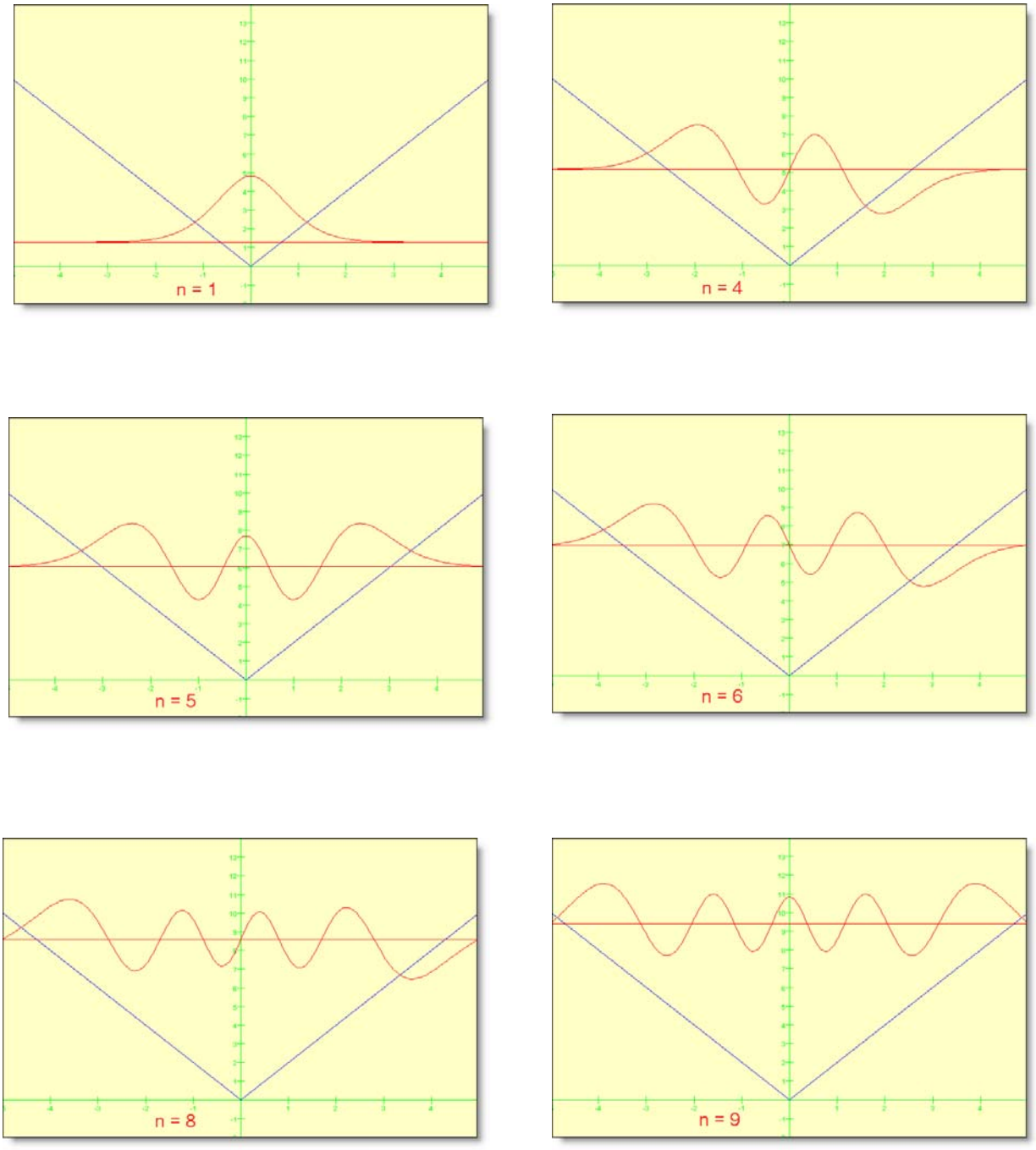
$$V = \frac{1}{2}x^2$$



รูปที่ 4.8 ฟังก์ชันคลื่นในบ่อศักย์แบบฮาร์มอนิก ออสซิลเลเตอร์ 1 มิติ

บ่อศักย์แบบสามเหลี่ยม Triangular well สมการของพลังงานศักย์เป็นดังนี้

$$V = 2.0|x|$$

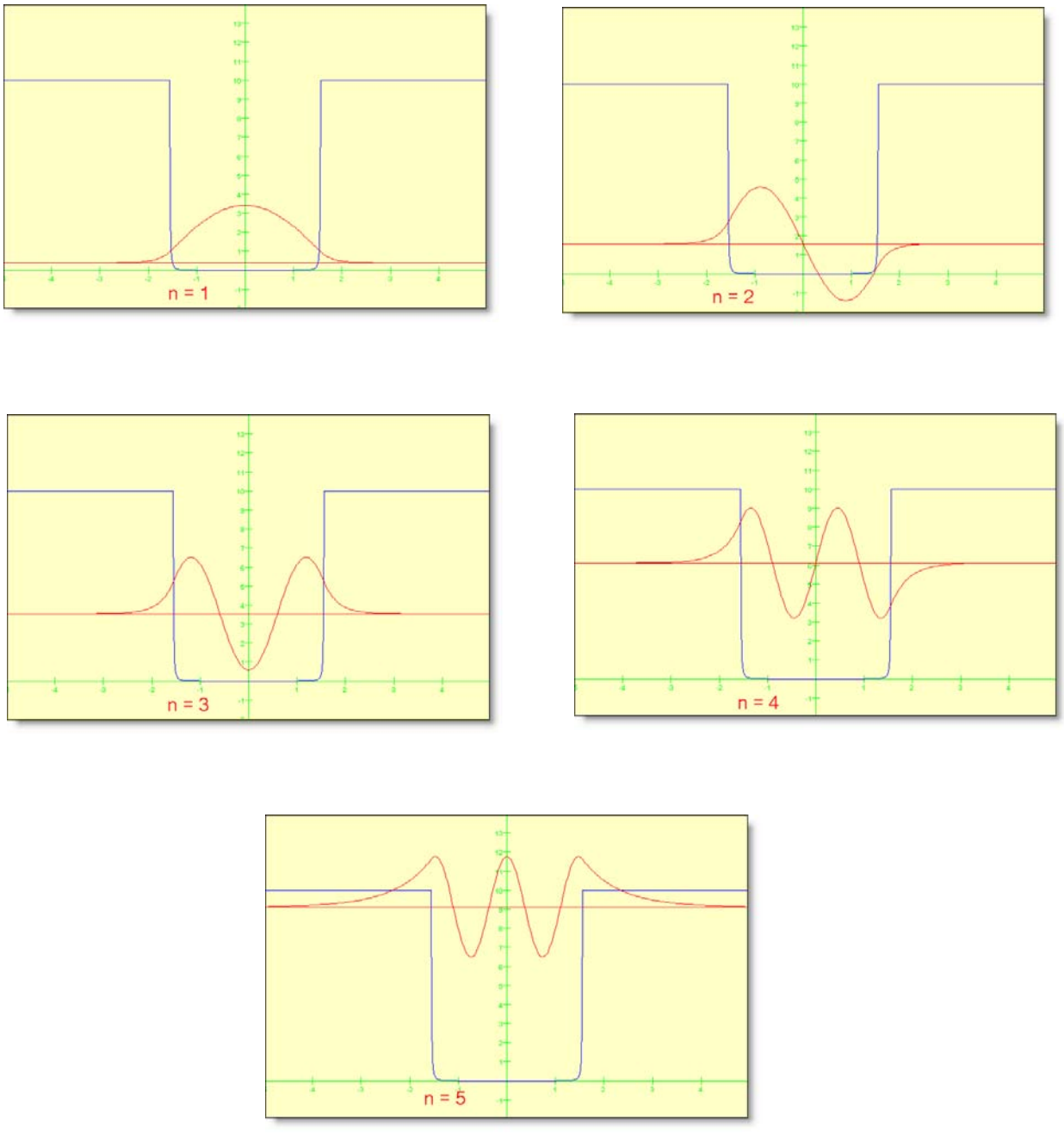


รูปที่ 4.9 ฟังก์ชันคลื่นในบ่อศักย์แบบสามเหลี่ยม

บ่อศักย์แบบ Square tangent well

$$V = \frac{1}{560}\tan^2 x$$

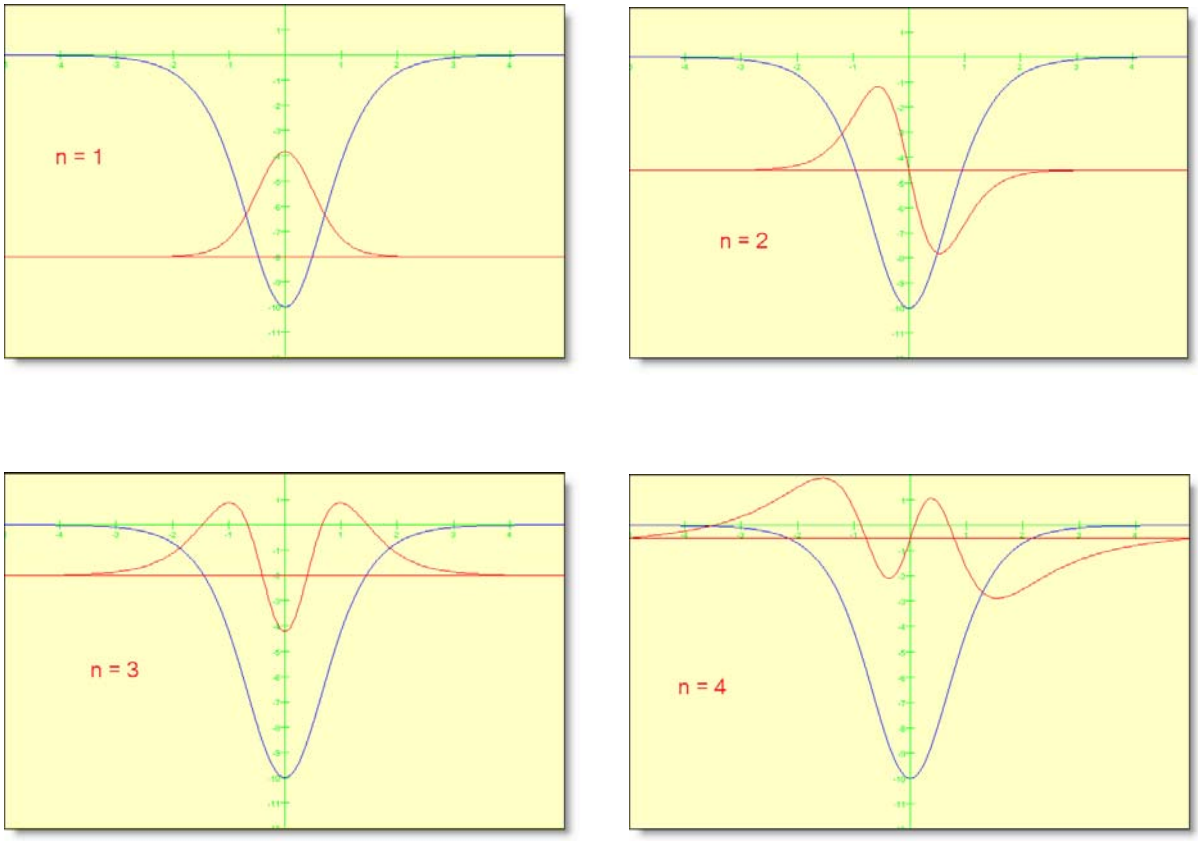
$$-\frac{\pi}{2} \le x \le \frac{\pi}{2}$$



รูปที่ 4.10 ฟังก์ชันคลื่นในบ่อศักย์แบบ Square tangent

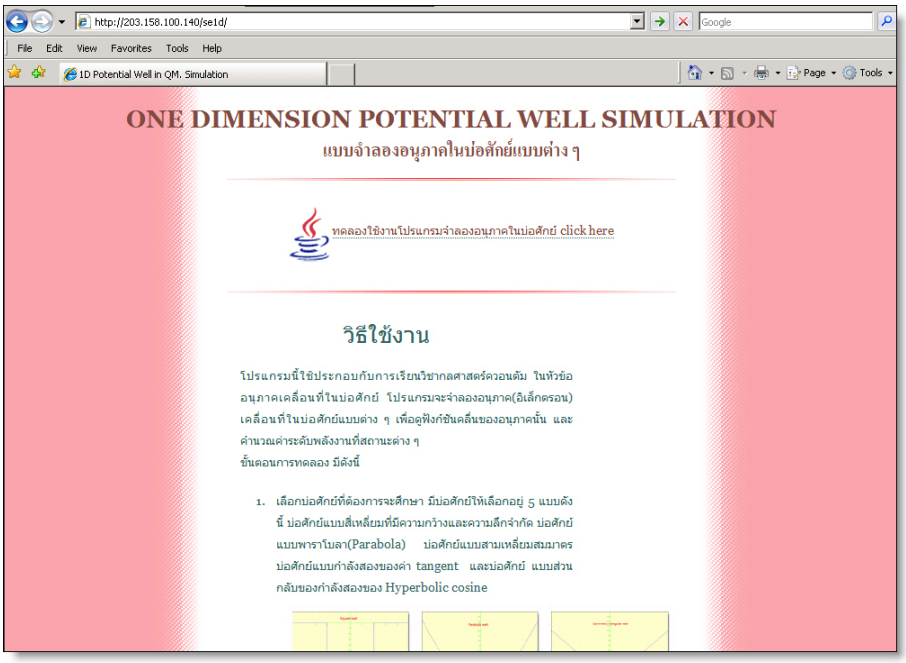
บ่อศักย์แบบ inverse square hyperbolic cosine

$$V = \frac{-10}{\cosh^2 x}$$



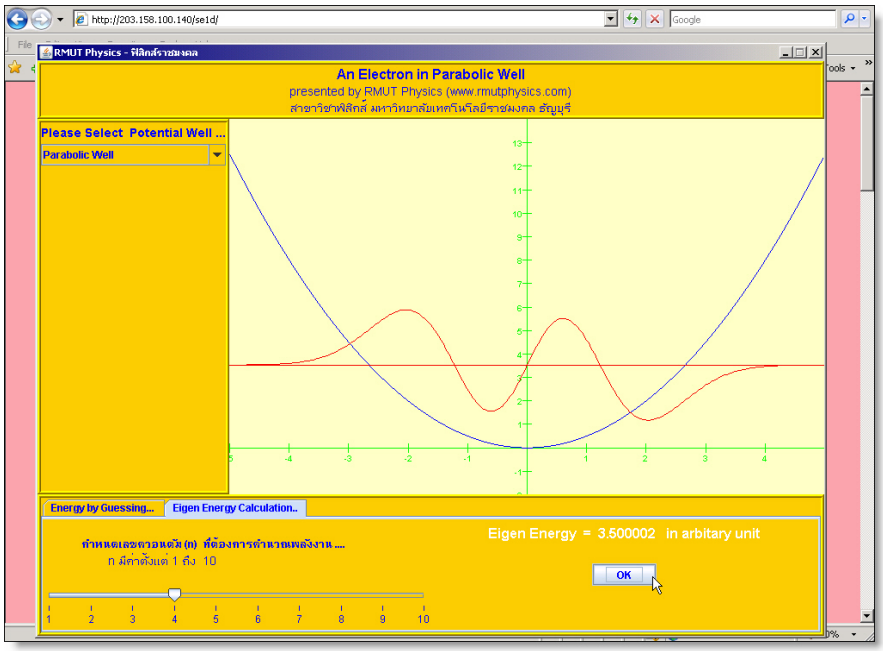
รูปที่ 4.11 ฟังก์ชันคลื่นในบ่อศักย์แบบสามเหลี่ยม

หลังจากทดสอบ และตรวจสอบความถูกต้องของโปรแกรมแล้ว ได้นำโปรแกรมไปติดตั้งในแม่ข่ายอินเตอร์เน็ต โดยติดตั้งไว้ที่ www.electron.rmutphysics.com/SE1D เมื่อเชื่อมต่ออินเตอร์เน็ตจะได้เว็บเพจ ดังภาพ



รูปที่ 4.12 เว็บเพจ ที่ใช้ติดตั้งโปรแกรมจำลองอนุภาคในบ่อศักย์แบบต่าง ๆ

เมื่อคลิกที่ข้อความให้โปรแกรมจำลองอนุภาคในบ่อศักย์แบบต่าง ๆ ทำงานจะได้หน้าตาของโปรแกรมดังรูปที่ 4.13



รูปที่ 4.13 เมื่อให้โปรแกรมทำงานผ่านเครือข่ายอินเตอร์เน็ต

4.3 โปรแกรมต้นฉบับ (Source code)

เมื่อนำการจำลองอนุภาคในบ่อศักย์แบบต่าง ๆ แสดงผลเป็นกราฟิก และสามารถทำงานผ่านเครือข่ายอินเตอร์เน็ต จะได้โปรแกรมต้นฉบับ(Source code) จำนวน 17 ไฟล์ แยกเป็นประเภทดังนี้

1. ส่วนที่ใช้ในการคำนวณ สมการชเรอดิงเงอร์ แบบ 1 มิติ ได้แก่ Schrodinger1D.java สามารถหาค่าพลังงานที่เป็นค่าจำเพาะเจาะจง และฟังก์ชันคลื่นของอนุภาค จัดเป็นหัวใจของโปรแกรมในงานวิจัยครั้งนี้

2. ส่วนที่เป็นฟังก์ชันของบ่อศักย์ ได้แก่ SquareWell.java (บ่อศักย์แบบสี่เหลี่ยมความลึกจำกัด) Parabolic.java (บ่อศักย์ของฮาร์มอนิก ออสซิลเลเตอร์) Triangular.java (บ่อศักย์แบบสามเหลี่ยมสมมาตร) TanSquare.java (บ่อศักย์แบบแทนเจนต์กำลังสอง) และ InveseCosh.java (บ่อศักย์แบบไฮเปอบอลิก คอซกำลังสอง) เมื่อผู้ใช้เลือกศึกษาบ่อศักย์แบบใด ฟังก์ชันของบ่อศักย์เหล่านั้นจะถูกเรียกใช้โดย Schrodinger1D.java และนำฟังก์ชันทางคณิตศาสตร์ไปพล็อตบนจอภาพ

3. ส่วนที่เป็นcontainer หลักของโปรแกรม โดยใช้ SWING GUI และส่วนที่เป็น Input ของโปรแกรม ได้แก่ BasePanel.java เป็นกรอบหน้าต่างหลักที่บรรจุหน้าต่างอื่น ๆ ไว้ทั้งหมด LeftPanel.java ใช้ในการแสดงผล ComboBox สำหรับเลือกบ่อศักย์ InputPanel.java EnergyPanel.java QuantumStatePanel.java ใช้สำหรับป้อนข้อมูลเกี่ยวกับพลังงานและ สถานะควอนตัม ทั้งสามโปรแกรมนี้ถูกรวมไว้ใน FooterPanel.java และ HeaderPanel.java ใช้แสดงหัวเรื่องซึ่งจะเปลี่ยนไปตามชนิดบ่อศักย์ที่เลือก

4. ส่วนที่แสดงผลกราฟิก มี 2 ไฟล์ คือ PlotPanel.java จะเป็นหน้าต่างหลักในการแสดงภาพกราฟิกทั้งหมด ตั้งแต่เขียนรูปบ่อศักย์เป็นฉากหลัง เขียนแกน x และ y รวมทั้งมาตราส่วนของกราฟ เขียนรูปฟังก์ชันคลื่นเมื่อค่าพลังงานเปลี่ยนไป รวมทั้งการ update หน้าจอ เมื่อมีการเปลี่ยนแปลงชนิดบ่อศักย์ โดยมี SEPanel.java เป็นตัวกำหนดพารามิเตอร์ต่าง ๆ

5. ส่วนที่เป็นทางเข้าของโปรแกรม หรือเป็น main program คือ mainApp.java โปรแกรมจะเริ่มต้นทำงานที่โปรแกรมนี้

รายละเอียดของโปรแกรมทั้ง 17 ไฟล์ได้รวบรวมไว้ที่เดียวกันในภาคผนวก 4 โปรแกรมอาจมีการแก้ไข ปรับปรุงเล็ก ๆ น้อย ๆ จนกระทั่งนาทีที่ปิดเล่มรายงานการวิจัยฉบับนี้ ต้องการต้นฉบับโปรแกรมที่มีการ update ล่าสุด สามารถ download ได้ที่ www.electron.rmutphysics.com/se1d

บทที่ 5
สรุปและอภิปรายผล

การพัฒนาโปรแกรมเพื่อศึกษาแบบจำลองอนุภาคในบ่อศักย์แบบต่าง ๆ หลังจากได้ทดสอบการทำงานและแก้ไขข้อบกพร่องต่าง ๆ แล้ว สามารถสรุปและอภิปรายผลลัพธ์ที่ได้ดังนี้

5.1 สรุปผลการวิจัย

5.1.1 อนุภาคเคลื่อนที่ในบ่อศักย์แบบสี่เหลี่ยมที่มีความลึกจำกัด

อนุภาคมวล m เคลื่อนที่ในบ่อศักย์ที่มีความกว้างเท่ากับ 2a ความลึกของบ่อศักย์ V ดังนี้

$$V = \begin{cases} -V_0 & if \; -a < x < a \\ 0 & if \quad |x| \geq a \end{cases}$$

การคำนวณพลังงาน (E) ของอนุภาคในบ่อศักย์ที่ระดับพลังงาน n ใด ๆ โดยวิธีวิเคราะห์สามารถหาได้จาก สมการ (2.32)

$$2a\sqrt{\frac{2m}{\bar{h}^2}(V_0 - E)} = (n-1)\pi + 2\tan^{-1}\sqrt{\frac{E}{V_0 - E}}$$

ในหน่วยอะตอม สมการจะถูกลดรูปเหลือเพียง

$$2a\sqrt{2(V_0 - E)} = (n-1)\pi + 2\tan^{-1}\sqrt{\frac{E}{V_0 - E}}$$

สมการนี้นำไปใช้ในการคำนวณหาพลังงาน E เมื่อกำหนดค่า n =1, 2, 3 ... ให้ความกว้างของบ่อ เท่ากับ 4 หน่วย (a = 2) และความลึกของบ่อศักย์เท่ากับ 10 หน่วย

5.1.2 อนุภาคเคลื่อนที่ในบ่อศักย์แบบฮาร์มอนิก ออสซิลเลเตอร์

สมการของบ่อศักย์มีดังนี้ $V = \frac{1}{2}x^2$

การคำนวณพลังงาน (E) ของอนุภาคในบ่อศักย์ที่ระดับพลังงาน n ใด ๆ (n = 1, 2, 3, ...) โดยวิธีวิเคราะห์สามารถหาได้จาก สมการ (2.45)

$$E_{n-1} = \bar{h}\omega(n - \frac{1}{2})$$

ในหน่วยอะตอม สมการจะถูกลดรูปเหลือเพียง $(\bar{h} = \omega = 1)$

$$E_{n-1} = (n - \frac{1}{2})$$

ผลลัพธ์ที่ได้จากโปรแกรมจำลองอนุภาคในบ่อศักย์ กับค่าที่ได้จากการคำนวณโดยการแทนค่าลงในสมการตามทฤษฎี พร้อมกับความคลาดเคลื่อน มีดังนี้

สถานะ n	Square well		% ความคลาดเคลื่อน	Parabolic (Harmonic Oscillator)		% ความคลาดเคลื่อน
	ผลลัพธ์จากโปรแกรม	ผลลัพธ์จากสมการ		ผลลัพธ์จากโปรแกรม	ผลลัพธ์จากสมการ	
1	-9.752 280	-9.750697	0.016	+0.500	+0.05	0
2	-9.011 696	-9.005423	0.069	+1.500	+1.50	0
3	-7.786 852	-7.772970	0.178	+2.500	+2.50	0
4	-6.095 848	-6.071860	0.395	+3.500	+3.50	0
5	-3.977 063	-3.941455	0.903	+4.500	+4.50	0
6	-1.536 339	-1.491250	3.023	+5.500	+5.50	0
7	+0.369 385	Unbound state	-	+6.500	+6.50	0
8	+0.567 383	Unbound state	-	+7.502	+7.50	0.026
9	+1.154 297	Unbound state	-	+8.511	+8.50	0.129
10	+2.072 021	Unbound state	-	+9.536	+9.50	0.379

5.1.3 ผลลัพธ์ที่ได้จากบ่อศักย์แบบอื่น ๆ แสดงเฉพาะค่าที่ได้จากการคำนวณของโปรแกรมจำลองอนุภาคในบ่อศักย์แบบต่าง ๆ ไม่ได้แสดงค่าความคลาดเคลื่อน เพราะไม่มีการคำนวณตามทฤษฎีมาเปรียบเทียบ

สถานะ n	Triangular	Square tangent	Square hyperbolic cosine
	$V = 2.0 \mid x \mid$	$V = \frac{1}{560}\tan^2 x$ $-\frac{\pi}{2} \le x \le \frac{\pi}{2}$	$V = \frac{-10}{\cosh^2 x}$
1	+1.283 745	+0.397 526	−7.999 999
2	+2.945 831	+1.578 424	−4.999 999
3	+4.092 519	+3.507 880	−2.000 000
4	+5.150 50	+6.107 141	−0.498 165
5	+6.073 163	+9.112 893	+0.136 219
6	+6.957 181	+10.363 892	+0.488 270
7	+7.775 388	+10.466 563	+0.997 425
8	+8.593 503	+11.360 108	+1.639 515
9	+9.418 333	+11.729 493	+2.402 942
10	+10.318 601	+12.688 721	+3.281 116

5.2 อภิปรายผล

1. พลังงานที่เป็นค่าเจาะจงที่สถานะควอนตัม n ที่คำนวณได้จากบ่อศักย์แบบสี่เหลี่ยมที่มีความลึกจำกัดและบ่อศักย์แบบฮาร์มอนิก ออสซิลเลเตอร์ มีค่าใกล้เคียงกับที่คำนวณได้จากทฤษฎี การคำนวณของโปรแกรมยังคงใช้หน่วยอะตอม คือใช้มวลของอิเล็กตรอน และค่าคงที่ของพลังค์เท่ากับ 1 ยังไม่ได้มีการแปลงกลับให้เป็นหน่วย SI

2. บ่อศักย์ที่ใช้ในแบบจำลอง ยังคงถูกจำกัดเป็นบ่อศักย์ที่มีเงื่อนไข V(−x) = V(x) นอกจากนี้ยังไม่สามารถใช้บ่อศักย์แบบอื่นที่ให้ค่า singularity ในช่วงของ x ที่กำหนดให้ เช่น บ่อศักย์แบบคูลอมบ์ (Coulomb well) เป็นบ่อศักย์ที่เกิดจากแรงทางไฟฟ้าตามกฎของคูลอมบ์ บ่อศักย์แบบยูกาวา (Yukawa Well) เป็นบ่อศักย์ที่เกิดจากแรงแบบเข้ม (Strong interaction) ที่ยึดเหนี่ยวอนุภาคกับอนุภาคภายในนิวเคลียส ทั้งสองกรณีจะเกิด singularity ที่ x = 0 ซึ่งจะต้องแก้ปัญหาในการที่จะให้โปรแกรมคำนวณ และยังต้องแก้ปัญหาการเขียนกราฟที่ตำแหน่ง x = 0 นี้ด้วย

3. บ่อศักย์แบบอื่น ๆ ที่เหลือ ได้แก่บ่อศักย์แบบสามเหลี่ยมสมมาตร บ่อศักย์แบบแทนเจนต์กำลังสอง และบ่อศักย์ไฮเปอร์บอลิก โคไซน์กำลังสอง ยังไม่มีค่าที่ได้ตามทฤษฎีมาเปรียบเทียบ

5.3 ข้อเสนอแนะ

1. ในเรื่องบ่อศักย์แบบสี่เหลี่ยม ควรให้ผู้ใช้สามารถกำหนดความกว้าง และความลึกของบ่อ ควรกำหนดจำนวนบ่อที่ใช้ในการจำลองสถานการณ์ด้วย

2. ตรวจสอบความคลาดเคลื่อนในการคำนวณ หรือหา algorithm ที่ให้ความแม่นยำและความคลาดเคลื่อนน้อยกว่านี้ ค่าพลังงานที่คำนวณได้ที่ n ค่ามาก ๆ จะมีความคลาดเคลื่อนมากกว่าที่ n มีค่าน้อย ๆ ควรมีการแปลงหน่วยจากหน่วยอะตอมให้เป็นหน่วย SI เช่น หน่วยของระยะทาง และพลังงานควรมีหน่วยเป็นนาโนเมตร และอิเล็กตรอนโวลต์ ตามลำดับ

3. ควรเพิ่มขีดความสามารถให้ผู้ศึกษากำหนดฟังก์ชันของบ่อศักย์เองได้ตามความต้องการ

4. ควรพัฒนาโปรแกรมให้มีลักษณะเป็นมิตรกับผู้ใช้ (user friendly) สามารถใช้เมาส์เลือกพลังงาน หรือเลือกเลขควอนตัมได้ โดยที่ไม่ต้องพิมพ์ผ่านแป้นพิมพ์ สามารถบันทึกผลการจำลองสถานการณ์และพิมพ์ผลลัพธ์เก็บไว้ได้

บรรณานุกรม

1. Burden, Richard L. and Faires, Douglas J. 1993. **Numerical Analysis**. New York: PWS Publishing.
2. Falstad, P. "1 Dimension Quantum State Applet." [Online]. Available: http://www.falstad.com/ [Retrieved August 20, 2008].
3. Griffith, J. David. 1994. **Introduction to Quantum Mechanics.** New Jersey: Prentice–Hall, Inc.
4. Joffre, M. Basdevant, J. and Dalibard, J. "Wave Mechanics" [Online]. Available:http://www.quantum–physics.polytechnique.fr/. [Retrieved August 20, 2008].
5. Kahaner, David. and others. 1989. **Numerical Methods and Software**. New Jersey: Prentice–Hall,Inc.
6. Peleg, Y., Pinni, R. and Zaarur, E. 1998. **Schaum's Outline of theory and problems of Quantum Mechanics**. New York: McGraw Hill Book.
7. Phillips, A.C. 2003. **Introduction to Quantum Mechanics**. New York: John Willey & Sons, Inc.
8. Rae, A. 2002. **Quantum Mechanics – Modern Development**. London: Institute of Physics Publishing,.
9. Walker, Robert D. , 1987. **Numerical Methods for Engineers and Scientists**. Pennsylvania: Tab Book Inc.
10. Udal, A.. Reeder, R. 2006. **Comparison of methods for solving the Schrodinger equation for multiquantum well heterostructure application.** Proc. Estonian Acad. Sci. Eng.,12:246–261.
11. Biom, A. 2006. "Computer algorithms for solving the Schrodinger and Poisson equations." [Online]. Available:http://www.lu.se/. [Retrieved October 15, 2010].
12. Fowler, M. 2007. "Schrodinger's Equation in 1 D: Some Examples." [Online]. Available:http://Galileo.phys.virginia.edu/classes/51.mf1i.fall02/OneDimSchr.htm. [Retrieved March 4, 2011].
13. Horstmann, C. and Cornell, G. 2008. Core Java Vol 1 Fundamentals. California: Sun Microsystem, Inc.

14. Horstmann, C. and Cornell, G. 2008. Core Java Vol 2 Advanced Features. California: Sun Microsystem, Inc.

15. Topley, K. 2000. **Core SWING Advanced Programming**. New Jersey: Prentice–Hall, Inc.

16. Cole, B. Eckstein, R. and etc. 2002. **Java SWING.** 2nd edition. California: O'Reilly.

17. Adamson, C. and Marinacci, J. 2007. **SWING Hacks**. California: O'Reilly.

18. Gutz, S. 2000. **Up to Speed with SWING**. Greenwich: Manning Publication.

17. Raedt, H. and Michielsen, K. 2008. Quantum Mechanics. [Online]. Available: http://rugth30.phys.rug.nl/quantummechanics/. [Retrieved April 19, 2011].

18. Feagin, J. 1994. **Quantum Methods with MathematicA**. New York: Springer–Verlag, Inc.

19. Moyer, C. 2006. "Numerical Solution of the Stationary State Schrodinger Equation Using Transparent Boundary Conditions." Computing in Science & Engineering, May–June 2006, page 50–58.

20. Grinter, Roger. 2005. **The Quantum in Chemistry: An Experimentalist's View**. England: John Wiley & Sons, Ltd.

ภาคผนวกที่ 1

ค่าคงที่บางค่า และ หน่วยอะตอม (Atomic units)

ค่าคงที่ที่พบบ่อยๆ ในกลศาสตร์ควอนตัม

ค่าคงที่	สัญลักษณ์	ระบบ SI	หมายเหตุ
Vacuum velocity of light	c	$2.99792\times10^{8}\,ms^{-1}$	
Elementary charge	e	$1.60218\times10^{-19}C$	$4.80321\times10^{-10}\,esu$
Boltzmann's constant	k_B	$1.38066\times10^{-23}\,JK^{-1}$	
Planck's constant	h	$6.62608\times10^{-34}\,Js$	$6.62608\times10^{-27}\,erg\,s$
Planck's constant / 2π	$\bar{h}$	$1.05457\times10^{-34}\,Js$	
Avogadro's constant	N_A	$6.02214\times10^{23}\,\mathrm{mo}l^{-1}$	
Permittivity of free space	ε_0	$8.85419\times10^{-12}\,F\,\mathrm{m}^{-1}$	
Permeability of free space	μ_0	$4\pi\times10^{-7}\,NA^{-2}$	
Rest mass of electron	m_e	$9.10939\times10^{-31}kg$	
Rest mass of proton	m_p	$1.67262\times10^{-27}\,kg$	
Bohr magneton	μ_B	$9.27402\times10^{-24}\,A\,m^2$	$9.27402\times10^{-21}\,erg\ gauss^{-1}$
Nuclear magneton	μ_N	$5.05079\times10^{-27}\,A\,m^2$	$5.05079\times10^{-24}\,erg\ gauss^{-1}$

หน่วยอะตอม (Atomic units)

การใช้หน่วยในระบบ SI วัดขนาดและมวลของอนุภาคในอะตอม หรือวัดปริมาณอื่น ๆ ที่เกี่ยวกับอะตอม จะเป็นหน่วยวัดที่มีขนาดใหญ่ไป ไม่สะดวกในการเขียนและการคำนวณ และถ้านำไปใช้กับการคำนวณเชิงตัวเลขในคอมพิวเตอร์ การคำนวณที่เกี่ยวข้องกับตัวเลขที่มีค่าน้อยเกินไปหรือมากเกินไป จะทำให้เกิดความคลาดเคลื่อน และความคลาดเคลื่อนนี้จะถูกสะสมเพิ่มขึ้นเรื่อย ๆ ถ้าการคำนวณนั้นมีการทำซ้ำ หรือมีการวนรอบการคำนวณ ผลลัพธ์สุดท้ายที่ได้จะห่างไกลจากค่าแท้จริง เพื่อให้เกิดความสะดวกในการอธิบายและคำนวณในระดับอะตอม จึงมีการกำหนด Atomic unit (au หรือ a.u.) ขึ้นมาใช้งาน บางครั้งเรียกชื่อเต็มว่า Hatree atomic unit ตัวย่อ au นั้น ต้องระวังความหมายด้วย เพราะบางครั้งจะหมายถึง astronomical unit หรือ arbitrary unit หรือ absorbance unit ในบางกรณี

ในหน่วยอะตอมจะกำหนดค่าคงที่พื้นฐานต่อไปนี้ให้มีขนาด 1 หน่วย ดังตาราง

ค่าคงที่	สัญลักษณ์	ค่าที่ได้ในระบบ SI
Electron rest mass	m_e	$9.1093826(16)\times10^{-31}$ kg
Elementary charge	e	$1.60217653(14)\times10^{-19}$ C
reduced Planck's constant (angular momentum)	$\bar{h} = \frac{h}{2\pi}$	$1.05457168(18)\times10^{-34}$ J•s
Coulomb force constant	$\frac{1}{4\pi\varepsilon_0}$	$8.9875517873681\times10^{9}$ kg•m^3•s^{-2}•C^{-2}
	หรือ $4\pi\varepsilon_0$	$\frac{10^7}{c^2}$ A^2 / N $= 1.112650056053618\ldots\times10^{-10}$ F /m

เมื่อ c คือ ความเร็วของแสงในสุญญากาศ มีค่าแท้จริงหรือค่ามาตรฐาน เท่ากับ 2.99792458×10^8 m/s ส่วน $\bar{h}$, m_e และ e ยังคงไม่เป็นค่าที่แท้จริง ค่าของมันแปรเปลี่ยนไปได้ตามความละเอียดถี่ถ้วนของการทดลอง ส่วนค่า $4\pi\varepsilon_0$ นั้นจัดว่าเป็นค่าแท้จริงได้ เพราะค่าของมันขึ้นอยู่กับค่า c ซึ่งเป็นค่ามาตรฐานแล้ว

ในหน่วยอะตอม นิยาม ค่าคงที่ Fine structure (α) มีค่าดังนี้

$$\alpha = \frac{1}{4\pi\varepsilon_0}\frac{e^2}{\bar{h}c}$$

แทนค่า $4\pi\varepsilon_0 = \frac{10^7}{c^2}$ จะได้ $\alpha = \frac{e^2 c}{\bar{h}}\times10^{-7}\ N\ A^{-2}$

$$\alpha = 7.29735257\times10^{-3}$$

$$= \frac{1}{137.035999} \simeq \frac{1}{137}$$

ถึงตรงนี้จะเห็นได้ว่า ความเร็วของแสงในหน่วยอะตอม จะมีค่าเป็น $c = \frac{1}{\alpha} \simeq 137$

หน่วยของปริมาณทางฟิสิกส์หรือค่าคงที่อื่นๆ นอกเหนือจากตาราง จึงเป็นหน่วยอนุพัทธ์ (derived unit) ของหน่วยอะตอมเช่นระยะทาง ความเร็ว พลังงาน เวลา ฯลฯ

ในการวัดความยาว หน่วยอะตอมจะใช้วัดความยาวเป็นรัศมีอะตอมของโบร์ หรือ Bohr radius (a_0) ซึ่งเป็นค่ารัศมีของอิเล็กตรอนวงโคจรแรกของอะตอมไฮโดรเจน โดยที่

$$a_0 = \frac{4\pi\varepsilon_0\bar{h}^2}{m_e e^2} = \frac{\bar{h}}{\alpha m_e c}$$

$a_0 = 5.291772085936\times10^{-11} m$ = 0.5291772085936 Angstrom

การวัดพลังงาน จะมีหน่วยวัดเป็น Hartree หมายถึงพลังงานที่เกิดจากแรงผลักกันของประจุ 2 ประจุ ที่อยู่ห่างกันเป็นระยะรัศมีอะตอมของโบร์

$$E_H = \frac{m_e e^4}{(4\pi\varepsilon_0)^2 \bar{h}^2} = \frac{\bar{h}^2}{m_e a_0} = \alpha^2 m_e c^2$$

$$= \ 4.3597441775\times10^{-18}\,J \ = \ 27.2113846 \ \text{eV}$$

หน่วยอนุพัทธ์อื่น ๆ แสดงไว้ในตารางดังนี้

ปริมาณ	ชื่อหน่วย	สัญลักษณ์	สมการ	ค่าที่ได้ในระบบ SI
ความยาว	Bohr radius	a_0	$a_0 = \frac{4\pi\varepsilon_0\bar{h}^2}{m_e e^2} = \frac{\bar{h}}{\alpha m_e c}$	$5.291772085936\times10^{-11} m$
พลังงาน	Hartree	E_H	$E_H = \frac{m_e e^4}{(4\pi\varepsilon_0)^2\bar{h}^2} = \alpha^2 m_e c^2$	$4.3597441775\times10^{-18} J$
เวลา		τ_0	$\tau_0 = \frac{(4\pi\varepsilon_0)^2\bar{h}^3}{m_e e^4} = \frac{\bar{h}}{E_H}$	$2.41888432650516\times10^{-17} s$
ความถี่		ν_0	$\nu_0 = \tau_0^{-1}$	$4.13413737\times10^{16} Hz$
ความเร็ว		v_B	$v_B = \frac{e^2}{(4\pi\varepsilon_0)\bar{h}} = \alpha c = \frac{a_0}{\tau_0}$	$2.187691263373\times10^{6} m\,s^{-1}$
แรง		F_0	$F_0 = \frac{e^2}{4\pi\varepsilon_0 a_0^2} = \frac{E_H}{a_0}$	$8.238722514\times10^{-8} N$
อุณหภูมิ		T_0	$T_0 = \frac{E_H}{k_B}$	$3.157746455\times10^{5} K$
ความดัน		p_0	$p_0 = \frac{E_H}{a_0^3}$	$2.942191219\times10^{13} Pa$
สนามไฟฟ้า		E_0	$E_0 = \frac{e}{4\pi\varepsilon_0 a_0^2} = \frac{E_H}{a_0^3}$	$5.14220651\times10^{11} V\ m^{-1}$
ศักย์ไฟฟ้า		U_0	$U_0 = \frac{e}{4\pi\varepsilon_0 a_0}$	$27.211385\,V$
กำลัง		P_0	$P_0 = \frac{E_H}{\tau_0}$	$1.80237814\times10^{-1} W$
Energy flux density		I_0	$I_0 = \frac{P_0}{a_0^2}$	$6.643640931\times10^{19}\ W\ m^{-2}$

ในการเขียนปริมาณต่าง ๆ ในหน่วยอะตอม เช่น มวลของอนุภาคมีขนาดเป็น 1.5 เท่าของมวลอิเล็กตรอน สามารถเขียนได้เป็น $m = 1.5\ m_e$ เป็นการเขียนที่ชัดเจน โดยแสดงหน่วยอะตอมเป็นสัญลักษณ์ m_e หรือจะเขียนเป็น $m = 1.5$ a.u. (a.u. หมายถึง “expressed in atomic unit หรือ

แสดงในหน่วยอะตอม”) หรือเขียนสั้น ๆ ว่า m = 1.5 ทั้งสองกรณีหลังนี้จะต้องบอกให้ชัดเจนว่าเป็นการระบุหน่วยในหน่วยอะตอม

การใช้หน่วยอะตอม จะทำให้การอธิบายปรากฏการณ์ในระดับอะตอมได้ง่ายเพียงใด ดูได้จากการอธิบายสมบัติต่างๆ ของอิเล็กตรอนในอะตอมไฮโดรเจนของโบร์ ที่สถานะพื้น (Ground state) ดังนี้

	หน่วยอะตอม	หน่วย SI
รัศมีวงโคจรของอิเล็กตรอน	1	0.52918 Angstrom
ความเร็วของอิเล็กตรอนในวงโคจร	1	$2.187691263373\times10^{6}\,m\,s^{-1}$
โมเมนตัมเชิงมุมของอิเล็กตรอน	1	$1.05457\times10^{-34}\,Js$
คาบที่ใช้ในการโคจร 1 รอบ	2π	$2.41888432650516\times10^{-17}\,s$
พลังงานยึดเหนี่ยวหรือ ionization energy	$\frac{1}{2}$	13.595 eV
แรงคูลอมบ์ระหว่างอิเล็กตรอนและนิวเคลียส	1	$8.238722514\times10^{-8}\,N$
สนามไฟฟ้าที่เกิดจากนิวเคลียส	1	$5.14220651\times10^{11}\,V\,m^{-1}$

จะเห็นว่าเราจะได้ตัวเลขที่มีขนาดน้อยลง มองเห็นภาพอะตอมด้วยตัวเลขที่ชัดเจนมากขึ้น

สมการต่าง ๆ จะถูกเขียนให้กระชับขึ้น เช่น สมการชเรอดิงเงอร์ของอิเล็กตรอนในพลังงานศักย์ใดใน 1 มิติ ซึ่งมีรูปสมการเป็น

$$\frac{\partial^2\psi(x)}{\partial x^2}+\frac{2m}{\bar{h}^2}\big(E-V(x)\big)\psi(x)=0$$

จะเหลือสั้น ๆ เพียง

$$\frac{\partial^2\psi(x)}{\partial x^2}+2\big(E-V(x)\big)\psi(x)=0$$

สามารถนำไปใช้ในการคำนวณเชิงตัวเลขได้สะดวกกว่าแบบเดิมเป็นอย่างมาก

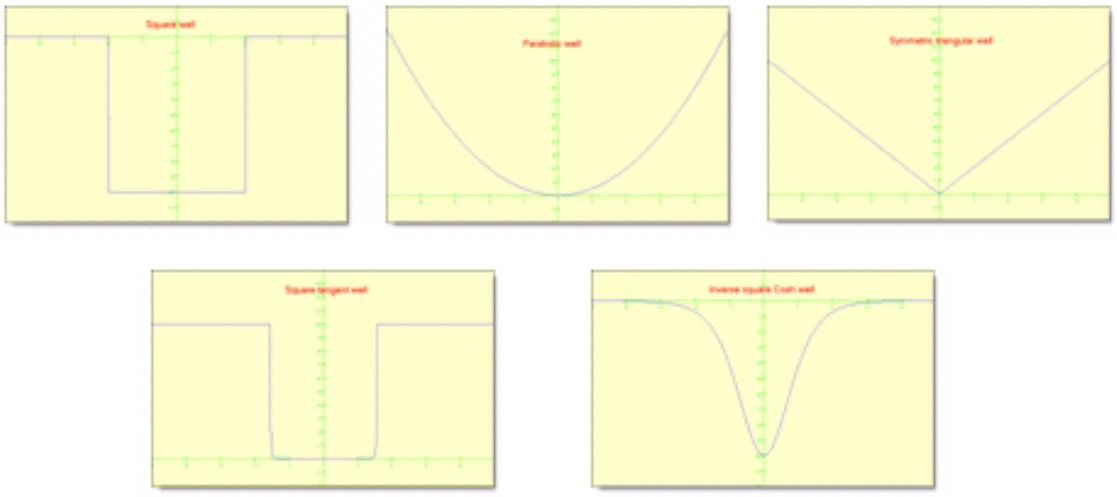
ภาคผนวกที่ 2
คู่มือการใช้งานโปรแกรมจำลองอนุภาคในบ่อศักย์แบบต่าง ๆ

ในการใช้งานโปรแกรมจำลองอนุภาคในบ่อศักย์แบบต่าง ๆ นี้ มีข้อสมมติฐานว่านักศึกษาได้เรียนรู้หรือกำลังเรียนรู้กลศาสตร์ควอนตัมเบื้องต้น ในวิชาฟิสิกส์ 2 หรือวิชากลศาสตร์ควอนตัมสำหรับนักศึกษาสาขาวิชาฟิสิกส์ ถึงหัวข้อสมการชเรอดิงเงอร์ เข้าใจคำว่าฟังก์ชันคลื่น ระดับพลังงานที่สถานะควอนตัมของอนุภาค

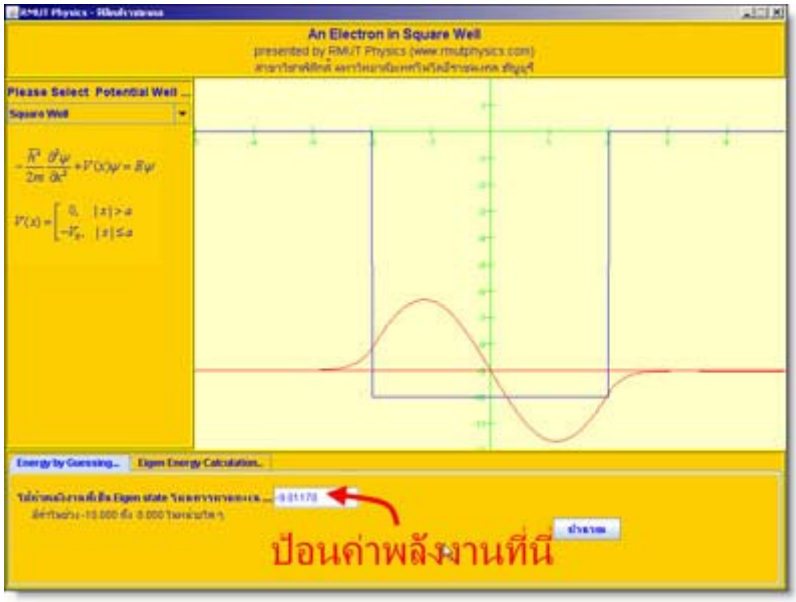
การนำโปรแกรมมาใช้ประกอบกับการเรียน ทำได้ดังนี้

1. เชื่อมต่อไปที่เว็บไซต์ www.electron.rmutphysics.com/SE1D สามารถเลือกการทำงานโดยรันโปรแกรมผ่านเครือข่ายอินเตอร์เน็ตหรือ ดาวน์โหลด SE1D.jar มาเก็บไว้ในเครื่องที่ผู้เรียนกำลังใช้อยู่

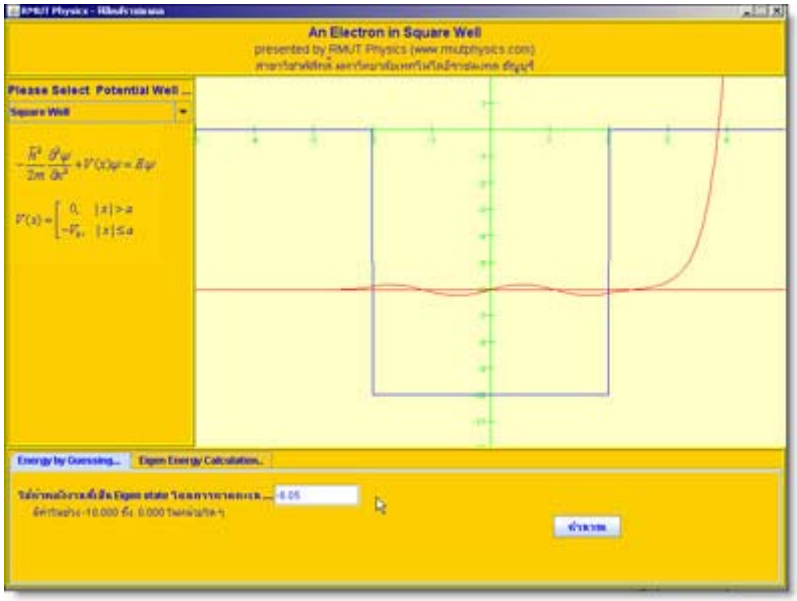
2. เลือกบ่อศักย์ที่ต้องการจะศึกษา มีบ่อศักย์ให้เลือกอยู่ 5 แบบดังนี้ บ่อศักย์แบบสี่เหลี่ยมที่มีความกว้างและความลึกจำกัด บ่อศักย์แบบพาราโบลา(Parabola) บ่อศักย์แบบสามเหลี่ยมสมมาตร บ่อศักย์แบบกำลังสองของค่า tangent และบ่อศักย์แบบส่วนกลับของกำลังสองของ Hyperbolic cosine

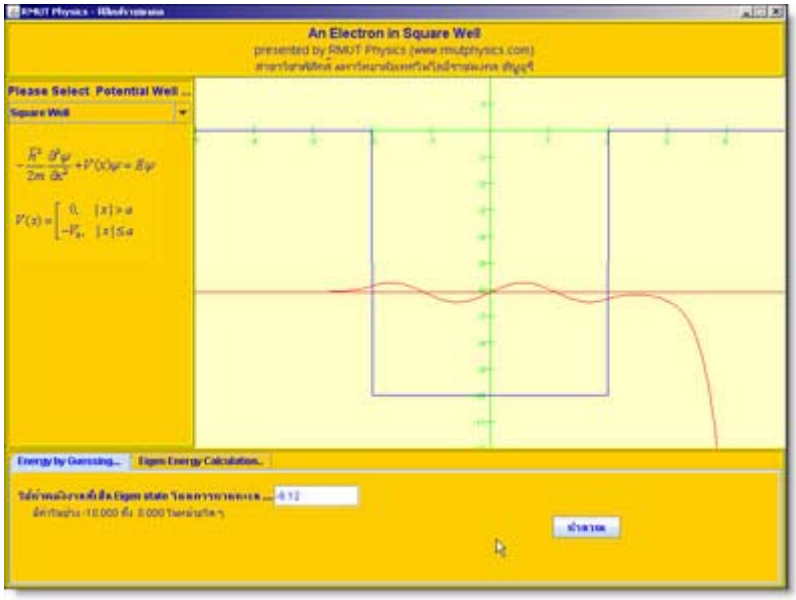


3. เมื่อเลือกบ่อศักย์ที่ต้องการแล้ว จะเห็นรูปกราฟของบ่อศักย์นั้นๆ ปรากฏบนจอด้านขวามือ ให้นักศึกษาทดลองใส่ค่าพลังงานของอนุภาคที่เป็นไปได้ที่อนุภาคยังคงอยู่ในบ่อศักย์นั้น แล้วคลิกที่ปุ่ม “คำนวณ” โปรแกรมจะแสดงฟังก์ชันคลื่นของอนุภาคที่ระดับพลังงานที่นักศึกษาเลือก ถ้าค่าพลังงานที่เลือกเป็นค่าที่เป็นไปได้ หรือเป็นค่า eigen value ฟังก์ชันคลื่นจะสอดคล้องกับเงื่อนไขขอบเขต นั่นคือ เส้นกราฟของฟังก์ชันคลื่น จะหายไปที่ผนังของบ่อศักย์ ด้านขวามือ หรือเส้นกราฟจะราบเรียบขนานกับแกน x



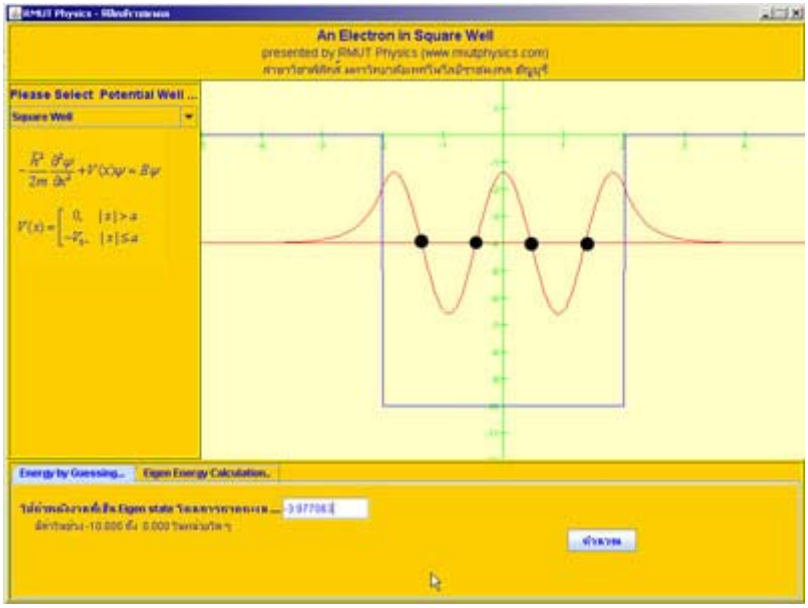
4. ถ้าฟังก์ชันคลื่นที่ได้ มี “หาง” ชี้ขึ้น หรือชี้ลงสู่ค่า infinity แสดงว่าค่าพลังงานที่คาดเดานี้ ยังไม่ใช่ระดับพลังงานที่อนุภาคสามารถมีได้ในบ่อศักย์นี้ ให้นักศึกษาใส่ค่าพลังงานค่าใหม่





5. ถ้าพลังงานที่ใส่เข้าไปครั้งแรก ทำให้หางของเส้นกราฟชี้ขึ้นที่ผนังบ่อด้านขวามือ และพลังงานที่ใส่เข้าไปครั้งที่สอง ทำให้หางของเส้นกราฟชี้ลงตรงข้ามกับครั้งแรก แสดงว่า ระดับพลังงานที่เป็น eigen value ต้องมีค่าอยู่ระหว่างค่าพลังงาน 2 ค่านี้ ให้นักศึกษาลดค่าหรือเพิ่มค่าจนกว่าจะได้รูปฟังก์ชันคลื่นที่เส้นกราฟด้านขวามือขนานกับแกน x

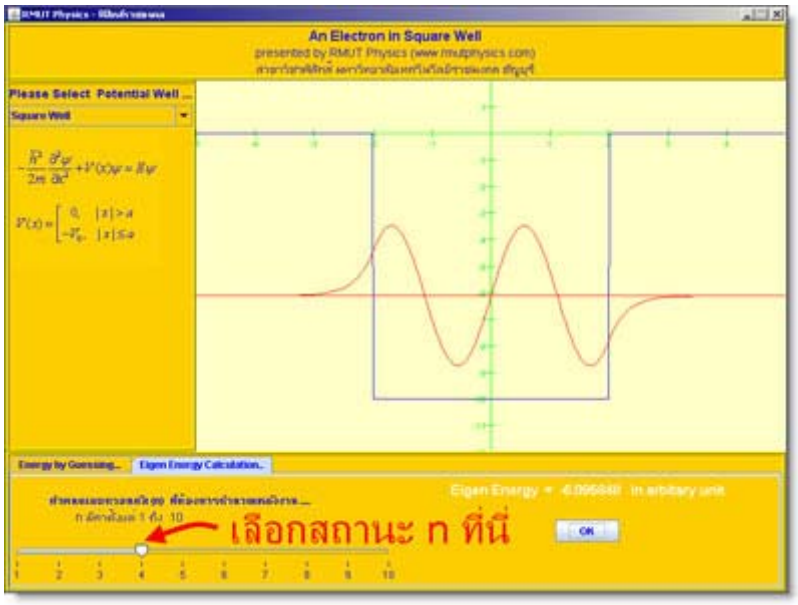
6. เมื่อได้รูปฟังก์ชันคลื่นตามที่ต้องการแล้ว ให้นับจำนวนครั้งที่เส้นกราฟของฟังก์ชันคลื่นตัดกับแกน x สถานะควอนตัม n ที่ระดับพลังงานนั้นหาได้จาก



สถานะควอนตัม n = จำนวนจุดของฟังก์ชันคลื่นที่ตัดแกน x + 1

จากรูป เส้นกราฟตัดกับแกน x จำนวน 4 จุด สถานะ n ของพลังงานที่ E = -3.977063 คือ 4+1 = 5

7. ต้องการทราบระดับพลังงานที่สถานะ n ต่าง ๆ (หลังจากเดาจนเหนื่อยแล้ว) ทำได้โดยคลิกที่แท็บ Eigen Energy Calculation เลือกสถานะ n ที่ต้องการทราบ โปรแกรมจะคำนวณค่าพลังงานที่เป็นค่า eigen และแสดงรูปฟังก์ชันคลื่นของอนุภาคที่ค่า n นี้ให้เห็นบนจอภาพ



8. อาจเลือกบ่อศักย์แบบอื่น ๆ สามารถศึกษาฟังก์ชันคลื่น และพลังงานที่เป็นค่าจำเพาะเจาะจง ได้ในลักษณะที่คล้ายกับที่กล่าวมาข้างต้น

9. เมื่อต้องการเลิกการใช้งาน ให้คลิกที่ปุ่ม กากบาท ที่มุมบนขวาของกรอบสี่เหลี่ยม โปรแกรมจะสิ้นสุดการทำงานทันที กรณีที่ทำงานผ่านเครือข่ายอินเตอร์เน็ต จะไม่มีคำสั่งหรือโปรแกรมใด ๆ ตกค้างอยู่ในฮาร์ดดิสก์ของเครื่องที่กำลังใช้งานอยู่เลย เมื่อต้องการจะศึกษาใหม่อีกครั้ง ให้นักศึกษาเข้าไปที่เว็บไซต์ดังที่กล่าวไว้ในข้อ 1. อีกครั้ง นักศึกษาสามารถเลือกเวลา สถานที่ (ที่สามารถเชื่อมต่อเน็ตได้) โดยไม่มีข้อจำกัด

ภาคผนวกที่ 3
การหารากสมการโดยวิธีแบ่งครึ่งช่วง และใช้ซอฟต์แวร์ Mathematica V 5.1

ในการเปรียบเทียบพลังงานของอนุภาค(E) ที่คำนวณได้จากโปรแกรมจำลองอนุภาคในบ่อศักย์กับค่าที่ได้จากสูตรที่ได้จากวิธีวิเคราะห์

$$2a\sqrt{2(V_0 - E)} = (n-1)\pi + 2\tan^{-1}\sqrt{\frac{E}{V_0 - E}} \qquad (1)$$

ในการคำนวณเพื่อเปรียบเทียบค่าในครั้งนี้ ได้กำหนดให้ a = 2 หน่วย (ความกว้างของบ่อ = 2a =4 หน่วย) ความลึกของบ่อศักย์ V_0 = –10 หน่วย (ตอนแทนค่าลงไปในสมการ (1) ไม่ต้องใส่เครื่องหมายลบ เพราะสมการ (1) นี้หามาจากการแทนค่า V_0 และ E ที่เป็นลบลงไปในสูตรเรียบร้อยแล้ว)

การหารากสมการ (E) ต้องใช้การคำนวณเชิงตัวเลข (Numerical method) ช่วยหาคำตอบ เพราะสมการในสูตรนี้จัดเป็น Transcendental function

ในการคำนวณเชิงตัวเลข จะใช้วิธีแบ่งครึ่งช่วง (Bisection method) หาคำตอบหรือหารากสมการ โดยเขียนเป็นโปรแกรมภาษาจาวา ทั้งหมด 5 ไฟล์ดังนี้

ไฟล์ที่ 1: CommonConstant.java

```
1:  /* File:CommonConstants.java
2:  * Project : Math Tools in Java
3:  * Author  : Wachara R.
4:  * First Released: 01 Dec 2006
5:  * Last Updated : May 02, 2011
6:  */
7:  package classes.MathTools.Common;
8:  public class  CommonConstants
9:  {
10:  /* Constants for calculating in All programs  */
11:          public static final double ZERO_APPROACH = 1.0e-7;
12:          // use 1.0e-7 for single precision
13:          // use 1.0e-16 for double precision
14:
15:          public static final double DEFAULT_TOLERANCE  = 1.0e-7;
16:
17:  // DEFAULT_INTERVAL for differential equation solver
18:          public static final double DEFAULT_INTERVAL  = 0.001;
19:
20:          //public static final int MAX_LOOP  = 50;
```

```
21:     public static final int MAX_LOOP  =100;
22:             public static final int ARRAY_SIZE=51; // Maximum equations
23:
24:     }
```

ไฟล์ที่ 2 : RootUtils.java

```
1:      /* File : RootUtils.java
2:      * Project: Math Tools in Java
3:      * Author : Wachara R.
4:      * First Released: 01 Dec 2006
5:      * First Updated : 13 Apr 2007 ( Thai new year day )
6:      *Last Updated : May 02, 2011
7:      */
8:
9:      package classes.MathTools.RootUtils;
10:
11:     import static classes.MathTools.Common.CommonConstants.*;
12:     import classes.MathTools.Common.Function;
13:     /** abstract class for finding root(s) of non linear equation */
14:     public abstract class RootUtils {
15:             Function function;
16:             private int counter;
17:             private int max_loop;
18:     /** constructor:
19:     *  @param f function for finding its root.
20:     *  @param max_loop : maximum number of loop for finding root
21:     */
22:     RootUtils ( Function f, int max_loop){
23:             this.function = f;
24:             this.max_loop = max_loop;
25:     }
26:     /** get the number of loop using for finding root
27:     *  @return  the number of iteration
28:     */
29:     public int getLoopCount() { return counter;}
30:     /** abstract method  for getting the root value */
31:     abstract double getRoot();
32:     /** reset counter for counting iteration */
33:     void resetCounter() {  counter = 0;              }
34:     /** test  for sure that the root is still between two point
35:     *  @param x1, x2   interval that the root is between
36:     */

37:     public void testInterval (double x1, double x2) throws      BadIntervalException {
38:             double y1 = function.Of(x1);
39:             double y2 = function.Of(x2);
```

```
40:                     if (y1*y2 > 0 || (x1 > x2)){
41:                     throw new BadIntervalException();
42:                     }
43:     }
44:     /** Checking if  number of loop exceed the limit */
45:     public void testLoopCount() throws ExceedLoopException {
46:                     counter = counter+1;
47:                     if (counter > max_loop){
48:                             throw new ExceedLoopException ();
49:                     }
50:     }
51:     /**  abstract method for finding the next better value of root */
52:     abstract void findNextPosition();
53:     /** abstract method for doing iteration for finding  root  */
54:     abstract void doIteration(int count);
55:     /** test whether the root has been calculated */
56:     abstract boolean IsSuccessed ();
57:     /** gathering the process of finding root in here */
58:     public boolean DoProcess() throws ExceedLoopException {
59:                     testLoopCount();
60:                     doIteration(counter);
61:                     findNextPosition();
62:                     return IsSuccessed();
63:     }
64:     }
65:
```

ไฟล์ที่ 3: Bisection.java

```
1:      /* File : Bisection.java
2:      * Project: Math Tools in Java
3:      * Author : Wachara R.
4:      * First Released: 01 Dec 2006
5:      * Last Updated : 20 Dec 2006
6:      */
7:      package classes.MathTools.RootUtils;
8:
9:      import static java.lang.Math.*;
10:
11:     import classes.MathTools.RootUtils.*;
12:     import classes.MathTools.Common.Function;
13:     import static classes.MathTools.Common.CommonConstants.*;
14:
15:     public class Bisection extends RootUtils{
16:                     private double xLeft;
17:                     private double xRight;
18:                     private double xMid;
```

```
19:                     private double f_xLeft;
20:                     private double f_xRight;
21:                     private double f_xMid;
22:
23:                     /** constructor
24:                     */
25:         public Bisection(Function function, double xMin, double xMax)
26:                                 throws BadIntervalException {
27:                                 super(function, MAX_LOOP);
28:                                 testInterval(xMin, xMax);
29:
30:                                 double yMin = function.Of(xMin);
31:                                 double yMax = function.Of(xMax);
32:
33:                                 if (yMin < 0){
34:                                             xLeft = xMin;
35:                                             xRight = xMax;
36:                                             f_xLeft = yMin;
37:                                             f_xRight = yMax;
38:                                 } else {
39:                                             xLeft = xMax;
40:                                             xRight = xMin;
41:                                             f_xLeft = yMax;
42:                                             f_xRight = yMin;
43:                                 }
44:                                 findNextPosition();
45:                                             try{
46:         /*
47:         System.out.println(" xLeft = "+ xLeft + " f(xLeft) = "+ f_xLeft);
48:         System.out.println(" xRight = "+ xRight + " f(xRight) = "+  f_xRight);
49:
50:         System.out.printf(" n\t  xLeft\t\tf(xLeft)    \txMid\t\tf(xMid)\t\txRight\t\tf(xRight)\n  ");
51:         System.out.printf("=========================================");
52:         System.out.printf("=========================================\n");
53:         */
54:                                             boolean IsOK;
55:                                             do{
56:                                                         IsOK = DoProcess();
57:         /*
58:         System.out.printf( " %d \t%f\t%f\t%f\t%f\n",getLoopCount(),
59:         xLeft,f_xLeft, xMid,
60:         f_xMid,xRight,f_xRight);
61:         */
62:                     }while (!IsOK);
63:         }// try
64:         catch (Exception e)     { System.out.println( e);}
```

```
65:         }
66:
67:         public double get_xLeft () { return xLeft;}
68:         public double get_xMid () { return xMid;}
69:         public double get_xRight () { return xRight;}
70:         public double get_f_xLeft () { return f_xLeft;}
71:         public double get_f_xRight () { return f_xRight;}
72:         public double get_f_xMid () { return f_xMid;}
73:         public double getRoot() { return get_xMid();}
74:
75:         void doIteration( int count) {
76:                         if ((f_xLeft*f_xMid) < 0 ){
77:                                         xRight = xMid;
78:                                         f_xRight = f_xMid;
79:                         } else {
80:                                         xLeft = xMid;
81:                                         f_xLeft = f_xMid;
82:                         }
83:
84:         }
85:         void findNextPosition() {
86:                         xMid = (xLeft + xRight)/2.0;
87:                         f_xMid = function.Of(xMid);
88:         }
89:
90:         boolean IsSuccessed() {
91:                         return abs(f_xMid) <= ZERO_APPROACH;
92:         }
93:
94:         }
95:
```

ไฟล์ที่ 4 : BisectionTest.java

```
1:      /* File : BisectionTest.java
2:      * Project: Math Tools in Java
3:      * Purpose: Test class for non linear equation solution.
4:      *
5:      * Author : Wachara R.
6:      * First Released: 2 May 2011
7:      * Last Updated : Mon 2 May 2011
8:      */
9:      package  JApp.SE1D;
10:
11:     import static java.lang.Math.*;
12:     import classes.MathTools.RootUtils.*;
13:     import classes.MathTools.Common.Function;
```

```
14:
15:     class  BisectionTest {
16:
17:     public static void main(String[] args) {
18:                             double a = 2.0;  // half width  of potential well
19:                             double  V = 10 ;  // depth of well
20:                             int n=1 ;  // qunatum number
21:                             for( n= 1; n < 7; n++) {
22:                             SWFunction swf = new SWFunction(n,a,V);
23:                             try{
24:                                     Bisection bs = new Bisection( swf, -10,10);
25:     System.out.printf("\n When n = " +n + " eigen Energy = "+ -          bs.getRoot());
26:                             }
27:
28:                             catch (Exception e)
29:                             { System.out.println(" Error(s) : "+ e);
30:                             }
31:             }
32:             }
33:
34:     }
```

ไฟล์ที่ 5: SWFunction.java

```
1:      /* File : SWFunction.java
2:      * Project: Schrodinger 1 D
3:      * Purpose: Test Bisection Algorithm.
4:      *
5:      * Author : Wachara R.
6:      * First Released: 2 May 2011
7:      * Last Updated : Mon 2 May 2011
8:      */
9:      package  JApp.SE1D;
10:
11:     import classes.MathTools.Common.Function;
12:
13:     public class SWFunction extends Function{  // SquareWell Function
14:             int quantumNumber;
15:             double halfWidth;
16:             double dept;
17:             double x;
18:
19:     SWFunction ( int n, double a, double d) {
20:                             quantumNumber = n;
21:                             halfWidth = a;
22:                             dept = d;
23:                             this.Of (x);
```

```
24:                                 }
25:                     public      double Of(double x) {
26:          return 2*halfWidth*Math.sqrt(2*(dept-x))-((double)quantumNumber
             -1.0)*Math.PI -  2.0*Math.atan(Math.sqrt(x/(dept-x)));
27:                     }
28:
29:         }
```

บรรทัดที่ 26 ตรงส่วนที่เป็นอักษรสีเข้ม คือสมการ (1) ที่ใช้ในการคำนวณหาค่า E เป็นสูตรที่ได้จากวิธีวิเคราะห์ ต้องการเปลี่ยนความกว้างและความลึกของบ่อศักย์ให้แก้ไข ค่า a และ V ตรงบรรทัดที่ 18, 19 ในไฟล์ที่ 4 BisectionTest.java

แปลโปรแกรมต้นฉบับให้เป็น ByteCode โดยใช้คำสั่งดังในภาพต่อไปนี้ พยายาม compile ให้ไฟล์มีการเรียงลำดับดังรูป และระวังชื่อโฟลเดอร์ที่ไฟล์นั้นอยู่ต้องถูกต้องด้วย

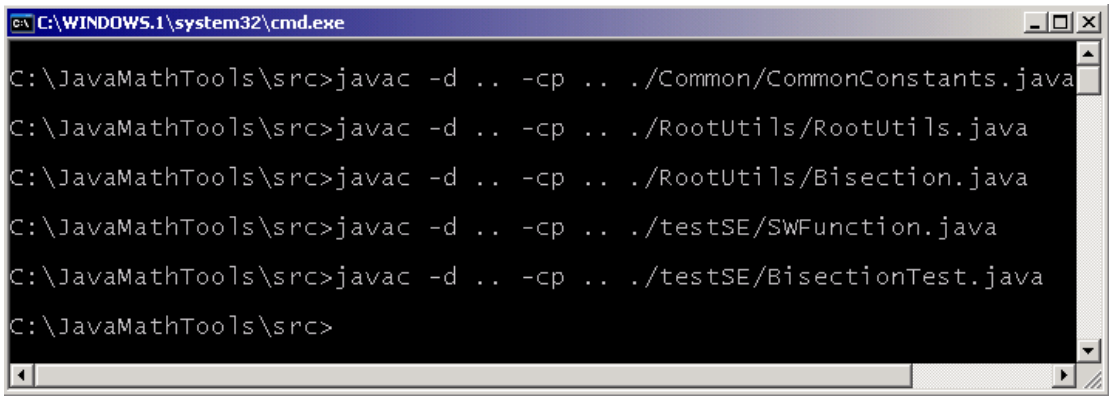
C:\JavaMathTools\src\Javac –d .. –cp .. ./Common/CommonConstants.java

C:\JavaMathTools\src\Javac –d .. –cp .. ./RootUtils/RootUtils.java

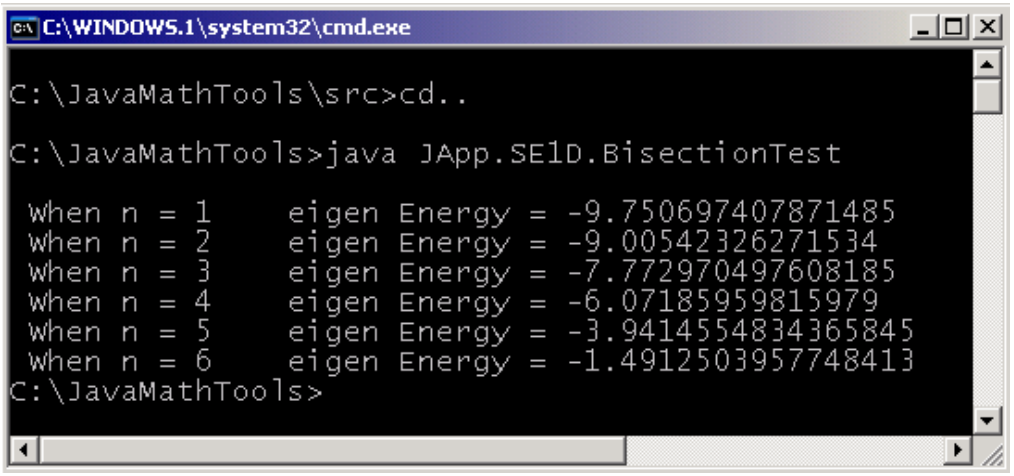
C:\JavaMathTools\src\Javac –d .. –cp .. ./RootUtils/Bisection.java

C:\JavaMathTools\src\Javac –d .. –cp .. ./testSE/SWFunction.java

C:\JavaMathTools\src\Javac –d .. –cp .. ./testSE/BisectionTest.java



ทดสอบการทำงานของโปรแกรม BisectionTest.class โดยใช้คำสั่งดังนี้



จะแสดงผลลัพธ์ที่ได้จากการคำนวณค่าพลังงานของอนุภาค เรียงลำดับตั้งแต่ n =1 ถึง 6

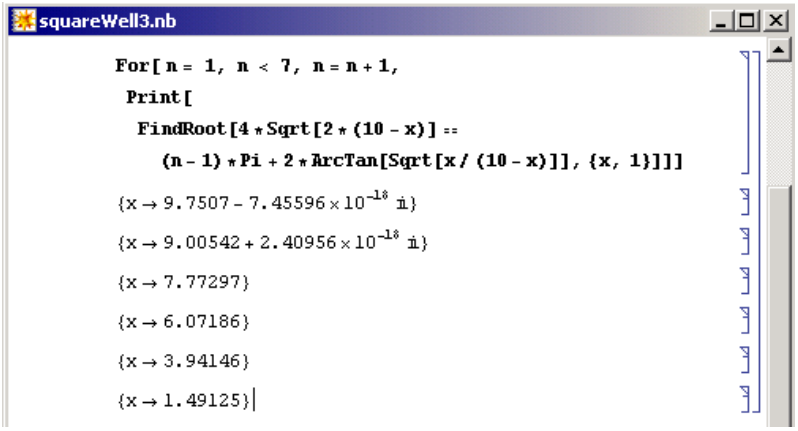
การใช้โปรแกรมสำเร็จรูป Mathematica version 5.1 หารากสมการ

เมื่อเปิดการใช้งานโปรแกรม Mathematica ให้พิมพ์คำสั่งลงไปในหน้าต่าง Notebook ของโปรแกรมดังต่อไปนี้

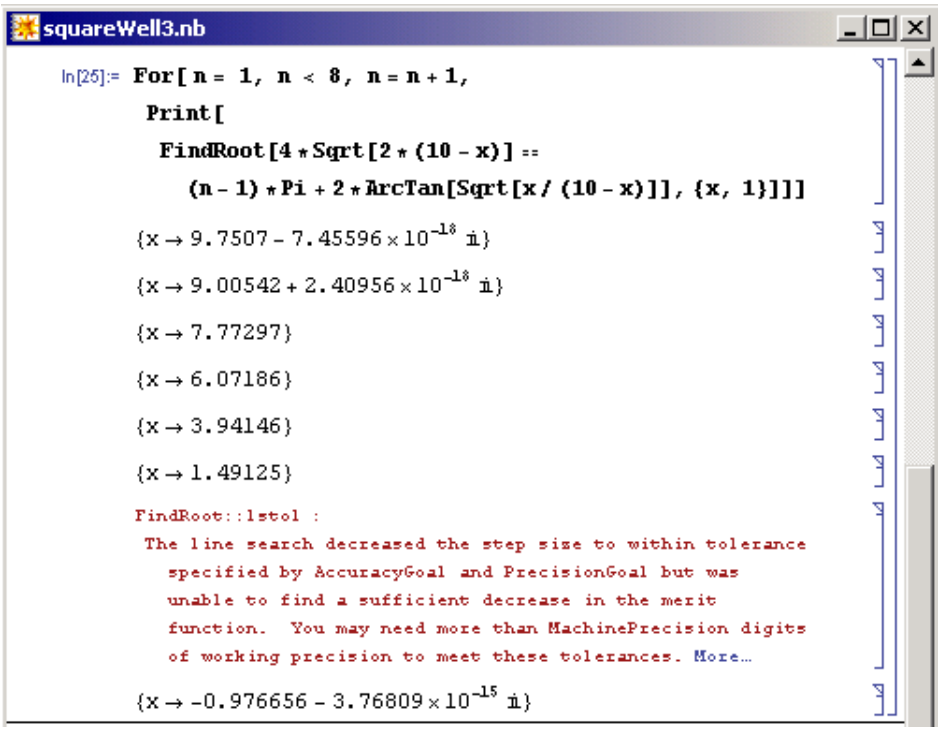
```
For[ n= 1, n < 7, n=n+1, Print[FindRoot[4*Sqrt[2*(10-x)]
        (n-1)*Pi+2*ArcTan[Sqrt[x/(10-x)]],{x,1}]]]
```

คำสั่งนี้หมายถึง ให้โปรแกรมทำงานวนรอบจำนวน 6 ครั้ง ตั้งแต่ n = 1, 2, 3, 4, 5, 6 แต่ละครั้งให้ค่า x (คือ E ในสูตรนั่นเอง) ที่ค่า n ต่าง ๆ กัน

เมื่อกดปุ่ม Shift + Enter พร้อมกัน โปรแกรมจะทำงาน แสดงผลลัพธ์ที่คำนวณได้ ดังรูป



เมื่อ n มีค่ามากกว่า 6 โปรแกรม mathematica จะแจ้งเตือนข้อความ ว่าพบข้อผิดพลาดในการคำนวณดังรูป



กรณีเช่นนี้พลังงาน E มีค่ามากกว่า พลังงานศักย์ของบ่อศักย์ อนุภาคไม่ได้อยู่ในสภาวะ Bound state อีกต่อไป

ภาคผนวกที่ 4
โปรแกรมต้นฉบับ (Source code)

โปรแกรมต้นฉบับที่ปรับปรุงล่าสุด (อังคาร 19 กรกฎาคม พ.ศ. 2554) เรียงตามลำดับดังนี้

1. ส่วนที่เป็นทางเข้าโปรแกรม mainApp.java และโปรแกรมที่กำหนดค่าคงที่ SEConstants.java, CommonConstants.java, Function.java
2. โปรแกรมที่ใช้คำนวณพลังงานและฟังก์ชันคลื่นของอนุภาค Schrodinger1D.java
3. โปรแกรมที่เป็นฟังก์ชันของบ่อศักย์ SquareWell.java, Quadratic.java, Triangular.java, TanSquare.java, InverseCosh.java
4. โปรแกรมที่เป็น panel สำหรับ comboBox และเป็น input ของโปรแกรม HeaderPanel.java, LeftPanel.java, FooterPanel..java , InputPanel.java, EnergyPanel.java และ QuantumStatePanel.java
5. โปรแกรมที่เป็นหลักในการแสดงผลกราฟิก SEPanel.java และ PlotPanel.java และ BasePanel.java

ไฟล์ที่ 1: mainApp.java

```
1:  /* File:mainApp.java
2:  * Project :Potential Well Simulation
3:  * Author  : Wachara R.
4:  * First Released: 19 Jan 2011
5:  * Last Updated :
6:  */
7:
8:  package JApp.SE1D;
9:
10: import java.awt.*;
11: import java.awt.event.*;
12: import javax.swing.*;
13: import classes.SE1D.*;
14:
15: public class mainApp extends JFrame
16: {
17:   private String title;
18:  private SEPanel displayPanel;
19:
20:     /**
21:      * Constructor.
22:      */
23: private mainApp( String title, SEPanel displayPanel)
```

```
24:         {
25:             this.title = title;
26:                                 this.displayPanel = displayPanel;
27:                                 setTitle(title);
28:                                 initFrame();
29:         }
30:         /**
31:          * Initialize the frame.
32:          */
33:     private void initFrame()
34:         {
35:             // Center the current frame.
36:     Dimension screenSize =
                                                Toolkit.getDefaultToolkit().getScreenSize();
37:                                 int width = screenSize.width*9/10;;
38:                                 int height = screenSize.height*9/10;
39:                                  setSize(width, height);
40:           setLocation((screenSize.width  – width )/2,
41:                         (screenSize.height – height)/2);
42:                       // ----------------- full screen version------------
43:                       //              setSize(screenSize.width,screenSize.height);
44:                       //              setLocation(0,0);
45:                       //-----------------------------------------------------------
46:             setLayout(new BorderLayout());
47:             add(displayPanel, BorderLayout.CENTER);
48:
49:             // Initialize the demo.
50:         //     displayPanel.panelInitialized();
51:
52:             // Window event handlers.
53:     /*
54:             addWindowListener(new WindowAdapter()
55:             {
56:                 public void windowOpened(WindowEvent ev)
57:                 {
58:                     repaint();
59:                 }
60:
61:
62:             });
63:     */
64:             // Resize event handler.
65:             addComponentListener(new ComponentAdapter()
66:             {
67:                 public void componentResized(ComponentEvent ev)
68:                 {
```

```
69:                     //  displayPanel.panelResized();
70:                    }
71:                 });
72:
73:         }
74:
75:         /**
76:          * Update the display without repainting the background.
77:          * @param g the graphics context
78:          */
79:         public void update(Graphics g)
80:         {
81:           //  displayPanel.panelDraw();
82:         }
83:         /**
84:          * Main.
85:          */
86:         public static void main(String args[])
87:         {
88:
89:                                   EventQueue.invokeLater(new Runnable()
90:             {
91:                public void run()
92:                {
93:       JFrame frame = new mainApp("RMUT Physics – ฟิสิกส์ราชมงคล",
                                    new SEPanel());
94:
95:                frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
96:                   frame.setVisible(true);
97:                                                frame.setResizable(false);
98:                }
99:             });
100:        }
101:    }
102:
```

ไฟล์ที่ 2: SEConstants.java

```
1:      /* File : SEConstants.java
2:      * Project : Schrodinger Eq in 1 Dimension
3:      * Author  : Wachara R.
4:      * First Released: June 26, 2011
5:      * Last Updated :  July 7, 2011
6:      */
7:      package classes.SE1D;
8:
9:      import java.awt.Color;
```

```
10:     public class  SEConstants
11:     {
12:     /* Constants for Graphics  */
13:
14:             private static final Color MAROON
                        = new Color(128, 0, 0);
15:             public static final Color BACKGROUND_COLOR = Color.orange;
16:             public static final Color FOREGROUND_COLOR = Color.blue;
17:
18:             public static final int NUMBER_OF_POINTS =300;
19:             public static final double MAX_X_VALUE=5.0;
20:             public static final  double MIN_X_VALUE= -5.0;
21:
22:             public static final int NO_ERROR = 0;
23:             public static final int MAXIMUM_NODE=20;
24:             public static final double SE_DEFAULT_TOLERANCE = 1.0e-3;
25:
26:     }
```

ไฟล์ที่ 3: CommonConstants.java

```
1:      /* File:CommonConstants.java
2:      * Project : Math Tools in Java
3:      * Author  : Wachara R.
4:      * First Released: 01 Dec 2006
5:      * Last Updated : May 02, 2011
6:      */
7:      package classes.MathTools.Common;
8:      public class  CommonConstants
9:      {
10:      /* Constants for calculating in All programs  */
11:             public static final double ZERO_APPROACH = 1.0e-7;
12:             // use 1.0e-7 for single precision
13:             // use 1.0e-16 for double precision
14:
15:             public static final double DEFAULT_TOLERANCE  = 1.0e-7;
16:
17:     // DEFAULT_INTERVAL for differential equation solver
18:             public static final double DEFAULT_INTERVAL  = 0.001;
19:
20:             //public static final int MAX_LOOP  = 50;
21:     public static final int MAX_LOOP  =100;
22:             public static final int ARRAY_SIZE=51; // Maximum equations
23:
24:     }
```

ไฟล์ที่ 4 Function.java

```
1: /*File : Function.java
2: * Project: Math Tools in Java
3: * by Wachara R.
4: * First Released: 01 Dec 2006
5: * Last Updated : 15 Dec 2006
6: */
7: package classes.MathTools.Common;
8: /** The mother class for function  with finding value of function
9: * and its deritvatives
10: *  @author Wachara R. ( Dec 14, 2006.)
11: */
12: public abstract class Function {
13: /** return the value of f(x)
14: * @param x  the value of x
15: * @return function of x
16: */
17: public abstract double Of(double x);
18: /** return the derivative of x
19: *       @param x  the value of x
20: *       @return derivative at x
21: */
22: public double derivativeOf(double x)
23 : { return 0.0;}
24: /** return the second derivative of x
25: *       @param x  the value of x
26: *       @return derivative at x
27: */
28: public double secondDerivativeOf(double x)
29:          { return 0.0;}
30 : }
```

ไฟล์ที่ 5: Schrodinger1D.java

```
1:          /**
2:          * Title: Schrodinger1D
3:          * Description compute energy, eigen state and Wave function
4:          *
5:          * @author: Wach R
6:          *@ version 0.1
7:          * Date : April 23, 2011
8:          * Last updated: May 2, 2011
9:          *============================
10:         * Schrodinger Eq :   d^2psi/dx^2 = 2*m/(hbar)^2(E-V)*psi
11:         *  define  k^2 = 2*m/(hbar)^2(E-V)
12:         * in atomic unit:  hbar = 1, m = 1
```

```
13:         */
14:          package classes.SE1D;
15:
16:          import static classes.MathTools.Common.CommonConstants.*;
17:          import static classes.SE1D.SEConstants.*;
18:          import classes.MathTools.Common.Function;
19:
20:
21:
22:         public class  Schrodinger1D
23:         {
24:
25:          int errorCode = 0;  // 0 -- means no error
26:
27:         double energy;
28:         double potentialMin;
29:         double potentialMax;
30:         double[] psi;  // wave function of the particle.
31:         double[] x;  // positions on x axis
32:         double xmin, xmax;  // min and max values of x
33:
34:         double tolerance = SE_DEFAULT_TOLERANCE;
35:         //   tolerance =1.0e-3  If less than 1.0e-3  there's some problem with  line 296
36:         double maxAmplitude;  // maximum amplitude of psi
37:         int firstPoint, lastPoint;
38:         double dx;
39:         Function potential;
40:
41:         double psiState ;   // keep current psi
42:         double  slopeState;  // keep d psi / dx
43:         double  xState;    // keep x value
44:
45:         public Schrodinger1D (Function potential, double xmin, double xmax, int numberOfPoints) {
46:                                     this.potential = potential;
47:                                     psi = new double[numberOfPoints];
48:                                     x= new double[numberOfPoints];
49:                                     this.xmin = xmin;
50:                                     this. xmax = xmax;
51:                                     dx = (xmax -xmin)/(numberOfPoints -1);
52:                                     firstPoint = 0;
53:                                     lastPoint = numberOfPoints;
54:                                     findMinAndMaxPotential();
55:         }
56:
57:         private void  findMinAndMaxPotential() {
58:         double xValue = xmin;
```

```
59:     double newPotential;
60:                             potentialMin = potential.Of(xValue);
61:                             potentialMax = potentialMin;
62:                              for(int i = 0; i <  psi.length; i++) {
63:                                                      x[i] = xValue;
64:                                                      newPotential = potential.Of(xValue);
65:                                                      if(potentialMin > newPotential)
                    potentialMin = newPotential;
66:                                                      if(potentialMax < newPotential)
                     potentialMax = newPotential;
67:                                                     xValue += dx;
68:          }
69:     }
70:     /**
71:        * finds  the area where the wave function is non-zero.
72:        * and  bounded by the firstPoint and lastPoint.
73:        *
74:        * @param energy
75:        */
76:       protected void findFirstAndLastPoint(double energy) {
77:         double  current_x = xmin;
78:                 double startX=0, stopX=0;
79:
80:                 if(( energy <= potentialMin) || (energy >= potentialMax))  {
81:                 System.out.println( "It's  not a bound state,
                    check your Potential and Energy ");
82:                 return;
83:                 }
84:
85:         // find the first point where E>V
86:
87:         for(int i = 0; i< psi.length;  i++) {
88:                                  if((energy - potential.Of(current_x))>0)  {
89:             firstPoint = i;
90:                                 startX = current_x;
91:             break;
92:           }
93:          current_x += dx;
94:         }
95:         current_x = xmax;
96:
97:         // find the last point where E>V
98:         for(int i = psi.length; i>firstPoint; i--) {
99:                                  if((energy - potential.Of(current_x)) > 0) {
100:                                lastPoint = i;
101:                                stopX= current_x;
```

```
102:                 break;
103:               }
104:               current_x -= dx;
105:             }
106:
107:         // Actually, wave function value is not zero immediately
108:         // it still decay at the both sides
109:         // Left side :
110:             double decay = 1;
111:             current_x = xmin+firstPoint*dx;
112:             while(decay > tolerance && firstPoint > 0) {
113:               firstPoint--;
114:               current_x -= dx;
115:                       // this region E < V  the result shoud be minus sign
116:             double k = Math.sqrt(2*Math.abs(energy-potential.Of(current_x)));
117:               decay *= Math.exp(-k*dx);
118:
119:
120:             }
121:                       // Right side:
122:             decay = 1;
123:             current_x = xmin+lastPoint*dx;
124:              // assume exponential decay on right hand side
125:              while(decay > tolerance && lastPoint < psi.length) {
126:                lastPoint++;
127:                current_x += dx;
128:             double k = Math.sqrt(2*Math.abs(energy-potential.Of(current_x)));
129:
130:               decay *= Math.exp(-k*dx);
131:
132:             }
133:           }
134:
135:           /**
136:            * Solves the Schroedinger Equation with the given energy.
137:            * @param energy double the energy
138:            * @return int node  ( number of zero crossing of wave function )
139:            */
140:         public int solveWaveFunction(double energy) {
141:             int node = 0;  // count the number of zero crossings.
142:             boolean slopeChanged = false;
143:             maxAmplitude = 0;
144:             double integral = 0;
145:                       double lastPsiState, lastSlopeState,lastXState;
146:                       double k2;   //  k^2 = 2*(E-V)
147:                       double  k2LastState, k2State;
```

```
148:                    double slope;
149:
150:                    this.energy = energy;
151:          findFirstAndLastPoint(energy);
152:          if(lastPoint – firstPoint <3) {
153:        // if number of point in range < 3 cannot use with Numerov method
154:            return 0;
155:          }
156:         // set wave function to zero outside the bound region
157:          for(int i = 0;i<firstPoint;i++) {
158:            psi[i] = 0;
159:            x[i] = xmin+i*dx;
160:          }
161:                    // state of the firstPoint
162:                    lastPsiState=0;
163:                    lastSlopeState = 0;
164:                    lastXState =xmin+firstPoint*dx;
165:                    psi[firstPoint]=lastPsiState;
166:                    x[firstPoint] = lastXState;
167:                    k2LastState = 2.0*(energy – potential.Of(lastXState));
168:                    // state of the firstPoint +1
169:                    psiState = dx;
170:          slopeState = 1.0;  // initial derivative of psi by dx
171:          xState =  xmin + (firstPoint+1)*dx;   // initial value of x
172:                    psi[firstPoint+1]=psiState;
173:                    x[firstPoint+1] = xState;
174:                    k2State = 2.0*(energy – potential.Of(xState));
175:                    slopeState = (psiState – lastPsiState)/dx;
176:        // find wave function (psi) with Numberov method
177:        // the differential eq must be in a form
178:        //                                   d^2 psi /dx^2  = –k^2 (psi)
179:        //                                    define  k^2 = 2*m/(hbar)^2(E–V)
180:        // ––––––––––––––––––––––––––––––––––––
181:        double  c =  dx*dx/12.0;  // coeficient for Numerov Algorithm
182:          for(int i = firstPoint+2; i<lastPoint; i++) {
183:                              x[i]= xmin+i*dx;
184:                              k2 = 2.0*(energy – potential.Of(x[i]));
185:                              psi[i] = ((2.0 –10.0*c*k2State)*psiState –
                        (1+ c*k2LastState)*lastPsiState)/(1+c*k2);
186:                              slope = (psi[i] – psiState)/dx;
187:
188:                              // keep previous values for next step
189:                    lastPsiState=psiState;
190:                    lastSlopeState = slopeState;
191:                    lastXState =xState;
192:                    k2LastState = k2State;
```

```
193:
194:          // Now store current values
195:                        psiState = psi[i];
196:                        k2State = k2;
197:                         xState = x[i] ;
198:                         slopeState = slope;
199:
200:               if(maxAmplitude<Math.abs(psiState)) {
201:                 maxAmplitude = Math.abs(psiState);
202:               }
203:         // Normalize  wave function  with trapezoidal rule
204:         //  Integrate f(x) dx =  deltaX ( summation fi – f0/2 –fn/2)
205:
206:                         integral += psiState*psiState;
207:
208:                        if(slopeState*lastSlopeState < 0)
209:                 slopeChanged = true;
210:
211:                          if(psiState*lastPsiState < 0)  {
212:                 node++;
213:                 slopeChanged = false;
214:               }
215:               if(Math.abs(psiState)>1.0e12) {  // prevent diverging so much
216:              for(int m = i+1;m<lastPoint; m++) { // fill remaining values
217:                   psi[m] = psiState;
218:                                        x[m] = xmin + m*dx;
219:                 }
220:                 break;
221:               }
222:             } // end of ----->  for(int i = firstPoint+2
223:
224:             integral *= dx;
225:         integral = integral –( psi[firstPoint]/2.0 + psi[lastPoint–1]/2.0)*dx;
226:                        // find the magnitude of wave function
227:                        double sqrtIntegral = Math.sqrt(integral);
228:
229:                        // fill the rest point at the right side
230:             for(int i = lastPoint; i < psi.length; i++) {
231:               psi[i] = psi[lastPoint–1];
232:               x[i] = xmin+i*dx;
233:             }
234:         /*
235:             // check to see if last value is close enough to a crossing
236:             if(slopeChanged&&(Math.abs(phi[phi.length–1])<=tolerance*maxAmp)) {
237:               crossing++;
238:             }
```

```
239:        */
240:
241:            normalize(sqrtIntegral);
242:
243:                        // check normalize it shoulld equals  1
244:         double sumOfPsi2 =0;
245:                        for(int i = firstPoint; i <lastPoint; i++) {
246:                          sumOfPsi2  += psi[i]*psi[i];
247:                        }
248:        //              System.out.println(" sum of  Psi ^2 = " + sumOfPsi2*dx );
249:            return node;
250:          }
251:        //  Normalize wave function
252:         private void normalize(double magnitude) {
253:            if(magnitude==0) {
254:              return;
255:            }
256:            for(int i = 0; i< psi.length; i++) {
257:              psi[i] /= magnitude;
258:              psiState /= magnitude;
259:             slopeState /= magnitude;
260:            }
261:            maxAmplitude /= magnitude;
262:          }
263:
264:          /**
265:           * Solves the eigen energy with the given energy level
266:           * @param int quantumNumber
267:           * @return int node  ( number of zero crossing of wave function
268:           */
269:        public double[ ] solveEigenEnergy( int quantumNumber)
270:        {
271:        double eMin;
272:        double eMax;
273:        int nodeCount=0;
274:        int counter = 0;
275:
276:                        if(quantumNumber > MAXIMUM_NODE)  {
277:                        System.out.println("Quantum  Number must not  greater than  "
                             + MAXIMUM_NODE);
278:                        return new double[psi.length];
279:        }
280:                        eMin = potentialMin;
281:                        eMax = eMin +1;
282:        while (counter < 20 && nodeCount <= quantumNumber) {
283:                        nodeCount = solveWaveFunction(eMax);
```

```
284:
285:// Increase energy until the number of node is greater than quantum number
286:                    eMax  = eMax +( eMax-eMin);
287:                    counter++;
288:        }
289:        counter = 0;
290:
291:        do{
292:                    double tempEnergy = (eMax + eMin)/2.0;
293:                    int crossing = solveWaveFunction(tempEnergy);
294:
295:
296:                    if((crossing == quantumNumber) && (Math.abs(psi[psi.length-1])
                                <= tolerance*maxAmplitude)) {
297:                                errorCode = 0;
298:
299:                                return psi;
300:                    }
301:                    if((crossing > quantumNumber) || ((crossing == quantumNumber)
                    &&(checkEvenNumber(crossing)*psi[psi.length-1] > 0)) ) {
302:                                eMax = tempEnergy;
303:                    304:
305:                    }
306:                    else {
307:                                eMin = tempEnergy;
308:
309:
310:                    }
311:                    counter ++;
312:        //          System.out.println();
313:        } while (counter < 32 && (eMax-eMin )> ZERO_APPROACH);
314:        errorCode = 1;
315:        return new double[psi.length];
316:        }
317:          /**
318:           * Gets the x coordinates
319:           */
320:          public double[] getXCoordinates() {
321:            return x;
322:          }
323:          public  double getDeltaX() {
324:                      return dx;
325:          }
326:          public double[] getWaveFunction() {
327:            return psi;
328:          }
```

```
329:        public double getEnergy() {
330:                        return energy;
331:        }
332:        public double getMaxAmplitude() {
333:                        return maxAmplitude;
334:
335:        }
336:        public double getMinPotential() { return  potentialMin;}
337:        public double getMaxPotential() { return potentialMax;}
338:         /**
339:           * Gets the error code from computation.
340:           * @return int
341:           */
342:          public int getError() {
343:            return errorCode;
344:          }
345:          private int checkEvenNumber(int number) {
346:                        if(number%2==0)
347:                                    return 1;  // Even
348:                        else
349:                                    return –1; //Odd
350:          }
351:        }
352:
```

ไฟล์ที่ 6 : SquareWell.java

```
 1:        /**
 2:        * Title: SquareWell.java
 3:        * Description Square well potential.
 4:        *
 5:        * @author: Wach R
 6:        *@ version 0.1
 7:        * Date : April 24, 2011
 8:        *============================
 9:         */
10:          package classes.SE1D;
11:
12:         import classes.MathTools.Common.Function;
13:
14:        public class SquareWell  extends Function {
15:                        double halfWidth;  // halfWidth of potential well
16:                        double dept;              //the depth of well  must be minus sign
17:        public          SquareWell( double halfWidth, double dept) {
18:                                    this.halfWidth = halfWidth;
19:                                    this.dept = dept;
20:                        }
```

```
21:
22:          public double Of(double x) {
23:                        if(  Math.abs(x) >= halfWidth)
24:                                    return 0;
25:                        else
26:                                                return  dept;
27:          }
28:
29:         }
30:
```

ไฟล์ที่ 7 : Quadratic.java

```
1:          /**
2:          * Title: Quadratic.java
3:          * Description Simple harmonic oscillator potential.
4:          *
5:          * @author: Wach R
6:          *@ version 0.1
7:          * Date : April 23, 2011
8:          *============================
9:           */
10:          package classes.SE1D;
11:
12:          import classes.MathTools.Common.Function;
13:
14:         public class Quadratic  extends Function {
15:           public double Of(double x) {
16:           return (x*x)/2;
17:           }
18:           public  double DerivativeOf(double x) {  return 0.0; }
19:
20:
21:         }
22:
```

ไฟล์ที่ 8 : Triangular.java

```
1:          /**
2:          * Title: Triangular.java
3:          * Description triangular potential.  V(x) = abs(x)
4:          *
5:          * @author: Wach R
6:          *@ version 0.1
7:          * Date : May 16, 2011
8:          *============================
9:           */
10:          package classes.SE1D;
```

```
11:
12:          import classes.MathTools.Common.Function;
13:
14:         public class Triangular  extends Function {
15:           public double Of(double x) {
16:           return Math.abs(x)*2.0;
17:           }
18:           public  double DerivativeOf(double x) {  return 1.0; }
19:
20:
21:         }
22:
```

ไฟล์ที่ 9 : TanSquare.java

```
1:          /**
2:          * Title: InverseCosh.java
3:          * Description triangular potential.  V(x) = abs(x)
4:          *
5:          * @author: Wach R
6:          *@ version 0.1
7:          * Date : May 16, 2011
8:          *Last Updated: July 3, 2011 : National Election days.
9:          *============================
10:          */
11:          package classes.SE1D;
12:
13:          import classes.MathTools.Common.Function;
14:
15:         public class TanSquare  extends Function {
16:           public double Of(double x) {
17:                       if(Math.abs(x) > 1.56)  return 9.995;
18:                       else
19:                                                 return((1.0/560.0)*Math.tan(x)*Math.tan(x) );
20:           }
21:
22:
23:         }
24:
```

ไฟล์ที่ 10 : InverseCosh.java

```
1:          /**
2:          * Title: InverseCosh.java
3:          * Description triangular potential.  V(x) = abs(x)
4:          *
5:          * @author: Wach R
```

```
 6:        *@ version 0.1
 7:        * Date : May 16, 2011
 8:        *============================
 9:         */
10:         package classes.SE1D;
11:
12:         import classes.MathTools.Common.Function;
13:
14:        public class InverseCosh  extends Function {
15:          public double Of(double x) {
16:        //  return (3-6.0/(Math.cosh(x)*Math.cosh(x)));
17:                        return(-10.0/(Math.cosh(x)*Math.cosh(x)) );
18:          }
19:        //  public  double DerivativeOf(double x) {  return 1.0; }
20:
21:
22:        }
23:
```

ไฟล์ที่ 11 : HeaderPanel.java

```
 1:        /**
 2:        * Title: HeaderPanel.java
 3:        * Description Simple harmonic oscillator potential.
 4:        *
 5:        * @author: Wach R
 6:        *@ version 0.1
 7:        * Date : April 23, 2011
 8:        *============================
 9:         */
10:        package classes.SE1D;
11:        import java.awt.*;
12:        import java.awt.event.*;
13:        import static classes.SE1D.SEConstants.*;
14:        import javax.swing.*;
15:        import javax.swing.border.Border;
16:
17:        /**
18:         * The header panel that displays title or any messages.
19:         */
20:        class HeaderPanel extends JPanel
21:        {
22:
23:           /** header label */ private JLabel label = new JLabel("",SwingConstants.CENTER);
24:        /** subHeader label */ private JLabel  sublabel =
                        new JLabel("presented by RMUT Physics (www.rmutphysics.com)"
25:                                                              ,SwingConstants.CENTER);
26:        /** subHeader label/ */ private JLabel sublabel2 = new JLabel
```

```
                ("สาขาวิชาฟิสิกส์ มหาวิทยาลัยเทคโนโลยีราชมงคล ธัญบุรี"
27:                             ,SwingConstants.CENTER);
28:         /**
29:          * Constructor.
30:          * @param headerText the header label text
31:          */
32:         HeaderPanel(String headerText)
33:         {
34:             this();
35:             label.setText(headerText);
36:                     Border raisedBevel = BorderFactory.createRaisedBevelBorder();
37:             Border loweredBevel = BorderFactory.createLoweredBevelBorder();
38:             Border compound = BorderFactory.createCompoundBorder
39:                             (raisedBevel, loweredBevel);
40:             setBorder(compound);
41:
42:         }
43:
44:         /**
45:          * Constructor.
46:          */
47:         private HeaderPanel()
48:         {
49:                             setLayout(new BorderLayout());
50:             setBackground(BACKGROUND_COLOR);
51:             label.setFont(new Font("Dialog", Font.BOLD, 16));
52:             label.setForeground(FOREGROUND_COLOR);
53:                             label.setBackground(BACKGROUND_COLOR);
54:                             label.setOpaque(true);
55:             sublabel.setFont(new Font("Dialog", Font.PLAIN, 14));
56:            sublabel.setForeground(FOREGROUND_COLOR);
57:                     sublabel.setBackground(BACKGROUND_COLOR);
58:                             sublabel.setOpaque(true);
59:
60:             sublabel2.setFont(new Font("Dialog", Font.PLAIN, 14));
61:            sublabel2.setForeground(FOREGROUND_COLOR);
62:                             sublabel.setOpaque(true);
63:
64:
65:                             add(label,BorderLayout.NORTH);
66:                             add(sublabel, BorderLayout.CENTER);
67:                             add(sublabel2, BorderLayout.SOUTH);
68:
69:         }
70:
71:         /**
```

```
72:              * Set the header label text in color.
73:              * @param text the text
74:              * @param color the color
75:              */
76:             void setLabel(String text, Color color)
77:             {
78:                 label.setForeground(color);
79:                 label.setText(text);
80:                                     label.setBackground(BACKGROUND_COLOR);
81:             }
82:          }
```

ไฟล์ที่ 12 : LeftPanel.java

```
1:          /* File:LeftPanel.java
 2:         * Project :Potential Well Simulation
 3:         * Author  : Wachara R.
 4:         * First Released: 19 Jan 2011
 5:         * Last Updated :  May, 05, 2011
 6:         */
 7:
 8:         package classes.SE1D;
 9:
10:         import java.awt.*;
11:         import java.awt.event.*;
12:         import java.awt.image.*;
13:         import java.io.*;
14:         import javax.imageio.*;
15:         import javax.swing.*;
16:         import javax.swing.border.Border;
17:         import javax.swing.plaf.ColorUIResource;
18:         //import  static classes.PWS.PWSConstants.*;
19:
20:
21:         /**
22:          * The left panel that displays potential functions or
23:          *  some parameters of program.
24:          */
25:         class LeftPanel extends JPanel
26:         {
27:            public static final Color BACKGROUND_COLOR = Color.orange;
28:                     public static final Color FOREGROUND_COLOR = Color.blue;
29:                     private JComboBox functionChoices;
30:                     private JLabel selectedLabel;
31:                     private  JPanel subPanel1;  // contain selected Label and  JCombobox
32:                     private String names[ ] = {"F1.png", "F2.png", "F3.png", "F4.png", "F5.png"};
33:                     private FunctionImagePanel imagePanel;
```

```
34:                 private String functionNames[] = {"Square Well","Parabolic Well", "Triangular Well", "Tangent Square Well",
35:                         "Inverse Hyperbolic  Cosine Well"};
36:                 private Graphics gr;
37:
38:                 private  int choice=1;
39:
40:
41:
42:         /** parent base panel */          private BasePanel   basePanel;
43:         /**
44:          * Constructor.
45:          * @param basePanel the parent for all panel
46:          */
47:         LeftPanel( BasePanel basePanel)
48:         {
49:             this();
50:             this.basePanel = basePanel;
51:                             subPanel1 = new JPanel();  //
52:                             subPanel1.setLayout( new GridLayout(2,1));
53:                             subPanel1.setBackground(BACKGROUND_COLOR);
54:                             imagePanel = new FunctionImagePanel(names);
55:
56:                 selectedLabel = new JLabel("Please Select  Potential Well ...");
57:                             selectedLabel.setVerticalAlignment(SwingConstants.CENTER);
58:
59:                             selectedLabel.setFont(new Font("Arial", Font.BOLD, 14));
60:                             selectedLabel.setForeground(FOREGROUND_COLOR);
61:                             subPanel1.add(selectedLabel);
62:
63:                             functionChoices = new JComboBox(functionNames);
64:                             functionChoices.setBackground(BACKGROUND_COLOR);
65:                             functionChoices.setForeground(FOREGROUND_COLOR);
66:                             subPanel1.add(functionChoices);
67:                             add(subPanel1, BorderLayout.NORTH);
68:                             add(imagePanel, BorderLayout.CENTER);
69:
70:
71:                             functionChoices.addItemListener(
72:                                     new ItemListener() {
73:                                     public void itemStateChanged(ItemEvent event) {
74:                                             if(event.getStateChange() ==
                    ItemEvent.SELECTED)                 {
75:                                     choice = functionChoices.getSelectedIndex();
76:                                     imagePanel.display(choice);
77:                                                     LeftPanel.this.basePanel.setHeaderLabel("An Electron in
                    "+functionNames[choice], FOREGROUND_COLOR);
```

```
78:
79:         LeftPanel.this.basePanel.setFunction(choice);
80:         double                  minEn = LeftPanel.this.basePanel.getMinEnergy();
81:         double      maxEn = LeftPanel.this.basePanel.getMaxEnergy();
82:         LeftPanel.this.basePanel.setMinAndMaxEnergy(minEn, maxEn);
83:         LeftPanel.this.basePanel.setEigenEnergyLabel("Eigen energy  = 0.0  in arbitary unit.",Color.white);
84:         LeftPanel.this.basePanel.setPsiFlag(false);
85:                                                         } // if
86:                                             }
87:                                 }
88:                                 );
89:             }
90:
91:             /**
92:              * Constructor.
93:              */
94:          LeftPanel()
95:             {
96:                 setBackground(BACKGROUND_COLOR);
97:                         Border raisedBevel = BorderFactory.createRaisedBevelBorder();
98:                 Border loweredBevel = BorderFactory.createLoweredBevelBorder();
99:                 Border compound = BorderFactory.createCompoundBorder
100:                                (raisedBevel, loweredBevel);
101:                setBorder(compound);
102:                                setLayout(new BorderLayout());
103:
104:            }
105:        int getChoice() {
106:                                return choice;
107:                    }
108:
109:        private class FunctionImagePanel extends JPanel
110:           {
111:        /** image buffer */ private BufferedImage[ ] bufferedImage=
                                                        new  BufferedImage[5];
112:
113:         /** buffer graphics context */  private Graphics gr;
114:
                                private int index=0;
115:                            private java.net.URL url ;
116:
117:        FunctionImagePanel(String[ ] fileName ) {
118:                    String folder ="JApp/SE1D/";
119:
120:
121:
```

```
122:
123:                    for(int i = 0 ; i <  fileName.length; i++) {
124:                    try{
125:
126:        url = this.getClass().getClassLoader().getResource(folder+fileName[i]);
127:
128:
129:
130:
131:                                            bufferedImage[i]= ImageIO.read(url);
132:
133:                                } catch (IOException e) { }
134:                    }
135:                    setBackground(Color.orange);
136:        }
137:        public void display( int i) {
138:                    index =i;
139:                    repaint();
140:        }
141:          public void paintComponent(Graphics g)
142:                {
143:                     super.paintComponent(g);
144:                     g.drawImage(bufferedImage[index],0,0,null);
145:                     g.drawString( " url = " + url, 10,20);
146:                }
147:             }
148:        }
```

ไฟล์ที่ 13: FooterPanel.java

```
1:          /**
2:          * Title: FooterrPanel.java
3:          * Description the lower Panel of base potential.
4:          *
5:          * @author: Wach R
6:          *@ version 0.1
7:          * Date : April 23, 2011
8:          *============================
9:           */
10:         package classes.SE1D;
11:         import java.awt.*;
12:         import java.awt.event.*;
13:         import static classes.SE1D.SEConstants.*;
14:         import javax.swing.*;
15:         import javax.swing.border.Border;
16:         import javax.swing.event.ChangeEvent;
```

```
17:     import javax.swing.event.ChangeListener;
18:
19:
20:     /**
21:      * The header panel that displays title or any messages.
22:      */
23:     class FooterPanel extends JPanel
24:     {
25:     private JTabbedPane tabPane = new JTabbedPane();
26:     private EnergyPanel  enPanel ;
27:     private QuantumStatePanel statePanel;
28:     private BasePanel basePanel;
29:     private int tabIndex =0;
30:         /**
31:          * Constructor.
32:          */
33:      FooterPanel(BasePanel basePanel)
34:         {
35:                                 this.basePanel =basePanel;
36:                                 setLayout(new BorderLayout());
37:             setBackground(BACKGROUND_COLOR);
38:                                  Border raisedBevel = BorderFactory.createRaisedBevelBorder();
39:             Border loweredBevel = BorderFactory.createLoweredBevelBorder();
40:             Border compound = BorderFactory.createCompoundBorder
41:                                 (raisedBevel, loweredBevel);
42:             setBorder(compound);
43:                                 tabPane.setBackground(BACKGROUND_COLOR);
44:                                 enPanel = new EnergyPanel(basePanel);
45:                                 statePanel = new QuantumStatePanel(basePanel);
46:                                 tabPane.addTab("<html><font color=blue> Energy by Guessing...</font></html>", enPanel);
47:                                 tabPane.addTab("<html><font color=blue>Eigen Energy Calculation..</font></html>", statePanel);
48:                                 add(tabPane, BorderLayout.CENTER);
49:                                 tabPane.addChangeListener(new ChangeListener() {
50:                                                         public void stateChanged(ChangeEvent evt) {
51:                                 JTabbedPane sourcePane = (JTabbedPane)evt.getSource();
52:                                                             // Get current tab
53:                                         tabIndex = sourcePane.getSelectedIndex();
54:
55:         }
56:     });
57:
58:         }
59:                     /*
60:                     double getInputEnergy() {
61:
62:                                 return enPanel.getInputEnergy();
```

```
63:                     }
64:                     */
65:                     void setMinAndMaxEnergyLabel(double min, double max) {
66:                             enPanel.setEnergyLimit(min, max);
67:                     }
68:                     public void setEigenEnergyLabel(String txt, Color color) {
69:                             statePanel.setEigenEnergyLabel(txt, color);
70:                     }
71:                     public  int getTabIndex() {
72:                             return tabIndex;
73:                     }
74:
75:          }
```

ไฟล์ที่ 14 : InputPanel.java

```
1:      /**
2:      * Title: InputPanel.java
3:      * Description : Parent class for input Panel.
4:      *
5:      * @author: Wach R
6:      *@ version 0.1
7:      * Date : May 29, 2011
8:      *============================
9:       */
10:     package classes.SE1D;
11:     import java.awt.*;
12:     import java.awt.event.*;
13:     import static classes.SE1D.SEConstants.*;
14:     import javax.swing.*;
15:     import javax.swing.border.Border;
16:
17:     /**
18:      * The header panel that displays title or any messages.
19:      */
20:     class InputPanel extends JPanel
21:     {
22:     JButton  calButton = new JButton("คำนวณ");
23:     JPanel  buttonPanel = new JPanel();
24:     JPanel controlPanel = new JPanel();
25:     JPanel blankPanel = new JPanel();
26:     JPanel calButtonPanel = new JPanel();
27:
28:         /**
29:          * Constructor.
30:          * @param headerText the header label text
31:          */
```

```
32:         InputPanel()
33:          {
34:                                 setLayout( new GridLayout(1,2));
35:                         setBackground(BACKGROUND_COLOR);
36:                                 setForeground(FOREGROUND_COLOR);
37:                                 calButton.setForeground(FOREGROUND_COLOR);
38:
39:                                 // GridLayout -- specified number of rows and columns and also
40:                                 //              with the specified horizontal and vertical gap.
41:                                 buttonPanel.setLayout(new GridLayout(3,1,10,10));
42:                                 controlPanel.setBackground(BACKGROUND_COLOR);
43:                                 blankPanel.setBackground(BACKGROUND_COLOR);
44:                                 buttonPanel.setBackground(BACKGROUND_COLOR);
45:                                 calButtonPanel.setBackground(BACKGROUND_COLOR);
46:                                 add(controlPanel);
47:                                 calButtonPanel.add(calButton);
48:                                 buttonPanel.add(blankPanel);
49:                                 buttonPanel.add(calButtonPanel);
50:                                 add(buttonPanel);
51:
52:
53:          }
54:
55:
56:
57:       }
```

ไฟล์ที่ 15 :EnergyPanel.java

```
1:      /**
 2:     * Title: EnergyPanel.java
 3:     * Description : Panel for input any guessing energy.
 4:     *
 5:     * @author: Wach R
 6:     *@ version 0.1
 7:     * Date : May 29, 2011
 8:     * Last Update: June 9, 2011
 9:     *============================
10:      */
11:     package classes.SE1D;
12:     import java.awt.*;
13:     import java.awt.event.*;
14:     import static classes.SE1D.SEConstants.*;
15:     import javax.swing.*;
16:     //import javax.swing.border.Border;
17:     import javax.swing.text.*;
18:     import java.text.DecimalFormat;
```

```
19:          import classes.MathTools.Common.Function;
20:
21:
22:         /**
23:          * The header panel that displays title or any messages.
24:          */
25:         class EnergyPanel extends InputPanel
26:         {
27:  JLabel  label1 = new JLabel("ใส่ค่าพลังงานที่เป็น Eigen state โดนการคาดคะเน ....");
28:          String str1 = "  มีค่าในช่วง ";
29:          JLabel label2 = new JLabel();
30:         private double minEn = –10.0;
31:         private double maxEn = 0;
32:         private double inputEnergy;
33:         JTextField  textField1 = new  JTextField("0.0");
34:         DecimalFormat decFormat;
35:
36:         //private BasePanel basePanel;
37:         private BasePanel basePanel;;
38:             /**
39:              * Constructor.
40:              * @param headerText the header label text
41:              */
42:         EnergyPanel(BasePanel basePanel)
43:             {
44:                                     super();
45:                                     this.basePanel = basePanel;
46:                                     controlPanel.setLayout(null);
47:                                     controlPanel.setBackground(BACKGROUND_COLOR);
48:                                     label1.setForeground(FOREGROUND_COLOR);
49:                                     label1.setBounds(10,15,300,25);  //
50:                                     controlPanel.add(label1);
51:                                     label2.setFont(new Font("Dialog", Font.PLAIN, 12));
52:                                     label2.setForeground(FOREGROUND_COLOR);
53:                                     label2.setBounds(20, 35,250,25);
54:                                     label2.setText(str1+(minEn)+ "ถึง " + maxEn+ "ในหน่วยใด ๆ ");
55:                                     controlPanel.add(label2);
56:                                     decFormat =new DecimalFormat("0.000");
57:
58:
59:                                     textField1.setForeground(FOREGROUND_COLOR);
60:                                     textField1.setBounds(310,15,100,25);
61:                                     controlPanel.add(textField1);
62:                                     textField1.setDocument( new  FloatFilter());
63:                                     textField1.addActionListener( new  ActionListener (){
64:                                     public void actionPerformed(ActionEvent e) {
```

```
65:                                         processCurrentEnergy();
66:                                         EnergyPanel.this.basePanel.doDrawing();
67:
68:
69:             }
70:         });
71:                 calButton.addActionListener(    new ActionListener() {
72:             public void actionPerformed(ActionEvent e) {
73:                                                                 processCurrentEnergy();
74:                                                                 EnergyPanel.this.basePanel.doDrawing();
75:             }
76:         });
77:
78:         }
79:                 public void setEnergyLimit(double minEn, double maxEn)
80:                 {
81:                         this.minEn = minEn;
82:                         this.maxEn = maxEn;
83:                 String      minE = decFormat.format(minEn);
84:                 String maxE = decFormat.format(maxEn);
85:                 label2.setText(str1+ minE +" ถึง  " + maxE+ " ในหน่วยใด ๆ ");
86:                 }
87:     public void processCurrentEnergy()
88:                 {
89:                         try  {
90:                         inputEnergy =  Double.parseDouble(textField1.getText());
91:                         if  (( inputEnergy < minEn) || (inputEnergy > maxEn)) {
92:                                                 EnergyPanel.this.basePanel.displayUserError("Your guessing energy  is out of
limitation");
93:                                                 return ;
94:                         }
95:                 EnergyPanel.this.basePanel.findWaveFunction(inputEnergy);
96:                         } catch ( Exception ev) {
97:                                                 EnergyPanel.this.basePanel.displayUserError("You SHOULD input any guessing
value of energy.....!");
98:
99:                         }
100:    }
101:
102:    class  FloatFilter extends PlainDocument
103:    {
104:    public static final String allowedCharacters ="0123456789.-";
105:
106:    public void insertString( int offset, String string, AttributeSet attr) throws BadLocationException{
107:                if(string == null) return;
108:                for(int i = 0; i <string.length(); i++) {
```

```
109:                                    if(allowedCharacters.indexOf(string.valueOf(string.charAt(i)))==-1)
110:                                            return;
111:                    } //for
112:                    if(string.indexOf(".") != -1) {
113:                                            if(getText(0,getLength()).indexOf(".") != -1) return;
114:                                    }
115:                    if(string.indexOf("-") != -1) {
116:                                    if(string.indexOf("-") != 0 || offset != 0) return;
117:                    }
118:                    super.insertString(offset, string, attr);
119:            }
120:
121:            } // end of subclass FloatFilter
122:
123:              }
```

ไฟล์ที่ 16: QuantumStatePanel.java

```
1:      /**
2:      * Title: QuantumStatePanel.java
3:      * Description : Panel for eigen energy calculation
4:      * at any quantum number.
5:      * @author: Wach R
6:      *@ version 0.1
7:      * Date : May 30, 2011
8:      *============================
9:       */
10:     package classes.SE1D;
11:     import java.awt.*;
12:     import java.awt.event.*;
13:     import static classes.SE1D.SEConstants.*;
14:     import javax.swing.*;
15:     import javax.swing.border.Border;
16:     //import javax.swing.JSlider;
17:     import javax.swing.event.ChangeEvent;
18:     import javax.swing.event.ChangeListener;
19:     import java.text.DecimalFormat;
20:
21:
22:     /**
23:      * The header panel that displays title or any messages.
24:      */
25:     class QuantumStatePanel extends InputPanel
26:     {
27:     private int minState = 1;
28:     private int maxState = 10;
29:     private int quantumState=1; // n = 1 at first start
```

```
30:         private double eigenEnergy = 0;
31:
32:          JLabel  label1 = new JLabel("                กำหนดเลขควอนตัม (n)  ที่ต้องการคำนวณพลังงาน ....");
33:          String str1 = "                      n มีค่าตั้งแต่ ";
34:          JLabel label2 = new JLabel();
35:          JLabel blankLabel = new JLabel( "    ");
36:
37:          JLabel energyLabel = new  JLabel("Eigen Energy  =  "+ eigenEnergy + "   in arbitary unit");
38:          JPanel labelPanel = new JPanel();
39:         JSlider slider = new JSlider (minState, maxState,1); // min, max, init value
40:
41:         private BasePanel basePanel;;
42:
43:         DecimalFormat decFormat;
44:
45:             /**
46:              * Constructor.
47:              * @param headerText the header label text
48:              */
49:         QuantumStatePanel(BasePanel basePanel)
50:             {
51:                                     super();
52:                                     this.basePanel = basePanel;
53:                                     energyLabel.setForeground(Color.white);
54:                                     energyLabel.setFont( new Font("Dialog",Font.BOLD, 16));
55:                                     blankPanel.add(energyLabel);
56:                                     controlPanel.setLayout(new BorderLayout());
57:                                     controlPanel.setBackground(BACKGROUND_COLOR);
58:                                     labelPanel.setLayout(new GridLayout(4,1));
59:                                     labelPanel.setBackground(BACKGROUND_COLOR);
60:                                     calButton.setText("     OK     ");
61:                                     calButton.addActionListener(     new ActionListener() {
62:                 public void actionPerformed(ActionEvent e) {
63:                                                                         QuantumStatePanel.this.basePanel.findEnergyAtQuantumState(
quantumState);
64:         eigenEnergy =QuantumStatePanel.this.basePanel.getEigenEnergy();
65:         setEigenEnergyLabel ("Eigen Energy  =  "+ decFormat.format(eigenEnergy )+ "   in arbitary unit", Color.white);
66:         // Sometime  "FindingEnergyAtQuantumState(n) doesnot return wave function
67:         // this following line could help.
68:         //QuantumStatePanel.this.basePanel.findWaveFunction(eigenEnergy);
69:          QuantumStatePanel.this.basePanel.doDrawing();
70:                 }
71:             });
72:                                     label1.setForeground(FOREGROUND_COLOR);
73:                                     labelPanel.add(blankLabel);
74:                                     labelPanel.add(label1);
```

```
75:                                 label2.setFont(new Font("Dialog", Font.PLAIN, 14));
76:                                 label2.setForeground(FOREGROUND_COLOR);
77:                                 label2.setText(str1+(minState) + " ถึง  " + maxState);
78:                                 labelPanel.add(label2);
79:                                 controlPanel.add(labelPanel, BorderLayout.NORTH);
80:
81:
82:                                 slider.setForeground(FOREGROUND_COLOR);
83:                                 slider.setFont(new Font("Dialog",Font.PLAIN,12));
84:                                 slider.setBackground(BACKGROUND_COLOR);
85:                                 slider.setMajorTickSpacing(1);
86:                                 slider.setPaintTicks(true);
87:                                 slider.setSnapToTicks(true);
88:                                 slider.setPaintLabels(true);
89:
90:         //              slider.setPreferredSize(new Dimension(50, 20));
91:             slider.addChangeListener(new ChangeListener() {
92:               public void stateChanged(ChangeEvent event) {
93:                 quantumState = slider.getValue();
94:               }
95:             });
96:                                 controlPanel.add(slider,BorderLayout.CENTER);
97:                                 decFormat =new DecimalFormat("0.000000");
98:           }
99:           public void setEigenEnergyLabel(String txt , Color color) {
100:
101:                                energyLabel.setForeground(color);
102:                                energyLabel.setText(txt);
103:
104:          }
105:
106:          }
```

ไฟล์ที่ 17 : SEPanel.java

```
1:          /* File:SEPanel.java
2:          * Project :Potential Well Simulation
3:          * Author  : Wachara R.
4:          * First Released: 19 Jan 2011
5:          * Last Updated :  June, 07, 2011
6:          */
7:
8:          package classes.SE1D;
9:
10:         import java.awt.*;
11:         import java.awt.event.*;
12:         import classes.MathTools.Common.Function;
```

```
13:     import static classes.SE1D.SEConstants.*;
14:     import javax.swing.*;
15:
16:     public class SEPanel extends BasePanel{
17:     int potentialChoice = 0;
18:      int numberOfPoints = NUMBER_OF_POINTS;
19:      int n = 1;
20:      double xmin = -5, xmax = +5;
21:      int widthPanel;
22:      int heightPanel;
23:      double  yFactor;
24:     Schrodinger1D se;
25:     double energy;
26:
27:
28:     public SEPanel()  {
29:     super("Potential Well  Simulation in Quantum Mechanics");
30:     se = new Schrodinger1D( new SquareWell(2.0, -10),xmin,
                                          xmax, numberOfPoints);
31:                 double[] psi = se.solveEigenEnergy(n);
32:         double[] x = se.getXCoordinates();
33:                 //widthPanel = 400; // getSize().width;
34:     //          heightPanel = 300; //getSize().height;
35:                 double maxAmplitude = se.getMaxAmplitude();
36:                 double energy = se.getEnergy();
37:                 yFactor = heightPanel/maxAmplitude;
38:                 }
39:
40:     }
41:
```

ไฟล์ที่ 18 : PlotPanel.java

```
1:      /* File:PlotPanel.java
 2:     * Project :Potential Well Simulation
 3:     * Author  : Wachara R.
 4:     * First Released: 02 June 2011
 5:     * Last Updated : 15 July , 2011
 6:     */
 7:
 8:     package classes.SE1D;
 9:
10:     import java.awt.*;
11:     import java.awt.event.*;
12:     import java.awt.image.*;
13:     import java.io.*;
14:     import javax.imageio.*;
```

```
15:         import javax.swing.*;
16:         import javax.swing.border.Border;
17:         import javax.swing.plaf.ColorUIResource;
18:         import static java.lang.Math.*;
19:         import static classes.SE1D.SEConstants.*;
20:         import classes.MathTools.Common.Function;
21:
22:
23:         /**
24:          * The panel that draws a set of axes, and plots points and lines.
25:          */
26:         class PlotPanel extends JPanel
27:         {
28:             private static final int TICK_SIZE = 5;
29:
30:             /** image buffer */              private Image        buffer;
31:             /** buffer graphics context */  private Graphics     bg;
32:             /** font metrics */             private FontMetrics fontMetrics;
33:
34:
35: /** label font */private Font charFont = new Font("Dialog",Font.BOLD, 10);
36:
37:                         /*  width of Plot panel */  int panelWidth;
38:                         int panelHeight;
39:                         /** x-axis row */           private int   xAxisRow;
40:             /** y-axis column */         private int   yAxisCol;
41:             /** minimum x value */      private double xMin;
42:             /** maximum x value */      private double xMax;
43:             /** minimum y value */      private double yMin;
44:             /** maximum y value */      private double yMax;
45:             /** x delta per pixel */    private double panelDeltaX;
46:             /** y delta per pixel */    private double panelDeltaY;
47:
48:
49:             /** parent graph panel */   private BasePanel basePanel;
50:
                        private Function currentFunction;
51:          /** Flag for  drawing wave function */  boolean  psiFlag = false;
52:          /** selected tab in Footer Panel */  int tabIndex = 0;
53:
54:         double[] psi;  // wave function of the particle.
55:         double[] xForPsi;  // positions on x axis
56:         double guessingEnergy;
57:         double eigenEnergy;
58:
59:             /**
```

```
60:          * Constructor.
61:          * @param BasePanel the parent base panel
62:          */
63:         PlotPanel(BasePanel basePanel)
64:         {
65:             this.basePanel = basePanel;
66:             setBackground(new Color (255, 255, 199));
67:                                 psi = new double[NUMBER_OF_POINTS];
68:                                 xForPsi= new double[NUMBER_OF_POINTS];
69:
70:         }
71:
72:                     public void setGraphProperties()  {
73:
74:                                 xMin                        =       PlotPanel.this.basePanel.getMinX();
75:                                             xMax    =  PlotPanel.this.basePanel.getMaxX();
76:                                             psiFlag = PlotPanel.this.basePanel.getPsiFlag();
77:                                             yMin       = PlotPanel.this.basePanel.getMinEnergy();
78:                                              yMax    = PlotPanel.this.basePanel.getMaxEnergy();
79:     if(yMax > 8)  { yMax = 14; yMin =-2; } else {  yMax = 2;yMin=-12; }
80:     //                System.out.println("y min = " + yMin + "        y max = " + yMax);
81:                                              panelWidth = getWidth();
82:                                             panelHeight = getHeight();
83:                                             panelDeltaX =  (xMax-xMin)/panelWidth;
84:                                             panelDeltaY       =     (yMax-yMin)/panelHeight;
85:
86:
87:                                             xAxisRow =   (int)Math.round( yMax/panelDeltaY);
88:                                             yAxisCol=       (int) Math.round(-xMin/panelDeltaX);
89:
90:                     //                      System.out.println("================\n\n");
91:                                             tabIndex = PlotPanel.this.basePanel.getTabIndex();
92:                                             if (psiFlag == true) {
93:                                              psi = PlotPanel.this.basePanel.getWaveFunction();
94:                                              xForPsi= PlotPanel.this.basePanel.getXCoordinates();
95:
96:
97:
98:                                             }
99:
100:                    }
101:
102:        public void paintComponent(Graphics bg)
103:        {
104:                        int pointCount;
105:                        super.paintComponent( bg);
```

```
106:                    setGraphProperties();
107:                    bg.setPaintMode();
108:                            bg.setFont(charFont);
109:                            fontMetrics= bg.getFontMetrics();
110:                             bg.setColor(Color.green);
111:                            drawAxes(bg);
112:                             bg.setColor(Color.blue);
113:                            drawPotential(bg);
114:                            if (tabIndex == 0 )
115:                guessingEnergy =PlotPanel.this.basePanel.getGuessingEnergy();
116:                else
117:                eigenEnergy = PlotPanel.this.basePanel.getEigenEnergy( );
118:
119:                             bg.setColor(Color.red);
120:                            drawWaveFunction(bg);
121:
122:        }
123:        void drawAxes(Graphics bg)
124:                {
125:                bg.drawLine(0, xAxisRow, panelWidth, xAxisRow);  //draw x axis
126:                        // X axis ticks
127:                        for ( int i =(int) round(xMin); i < round(xMax); i++) {
128:                                int column =(int)round((i-xMin)/panelDeltaX);
129:                                bg.drawLine(column, xAxisRow -TICK_SIZE, column,
                 xAxisRow+TICK_SIZE);
130:                                if(i !=0) {
131:                                        String number = Integer.toString(i);
132:                                        int w = fontMetrics.stringWidth(number);
133:                                        int x = column -w/2;
134:                                        int y = xAxisRow +TICK_SIZE +
                fontMetrics.getAscent();
135:                                        bg.drawString(number, x , y);
136:                                }
137:                        }
138:                // draw y axis
139:                        bg.drawLine(yAxisCol,0, yAxisCol, panelHeight);
140:                        // Y Axis ticks
141:                        for ( int i =(int) round(yMin); i < round(yMax); i++) {
142:                                int row =(int)round((yMax-i)/panelDeltaY);
143:      bg.drawLine(yAxisCol -TICK_SIZE, row, yAxisCol+TICK_SIZE,row);
144:                                if(i !=0) {
145:                                        String number = Integer.toString(i);
146:                                        int w = fontMetrics.stringWidth(number);
147:                                        int x = yAxisCol - TICK_SIZE- w;
148:                                        int y = row + fontMetrics.getAscent()/2;
149:                                        bg.drawString(number, x , y);
```

```
150:                                        }
151:                                } // for
152:                        }
153:        void drawPotential(Graphics bg)
154:                        {
155:        int  xScreen [ ]  = new int[panelWidth];
156:        int yScreen [ ]  = new int[panelWidth];
157:
158:            currentFunction =PlotPanel.this.basePanel.getCurrentFunction();
159:                                for(int column = 0; column  < panelWidth; column++) {
160:                                        double tempX = xMin +column*panelDeltaX;
161:                                        double tempY = currentFunction.Of(tempX);
162:                                 if(( tempY >= yMin ) && ( tempY <= yMax) ) {
163:                                 yScreen[column]= (int) round((yMax-tempY)/panelDeltaY);
164:                                                xScreen[column] = column;
165:                                                }
166:                                } // for column
167:                                 bg.drawPolyline( xScreen, yScreen, panelWidth-1);
168:                        }
169:        void drawWaveFunction(Graphics bg)
170:                        {
171:        int [ ] xScreen4Psi = new int[NUMBER_OF_POINTS];
172:        int [ ] yScreen4Psi = new int[NUMBER_OF_POINTS];
173:        double scaleFactor = 4;
174:        int yOffset;
175:
176:                        if( tabIndex== 0 )
177:         yOffset = (int) round((yMax -  guessingEnergy)/panelDeltaY) - xAxisRow;
178:                        else
179:         yOffset = (int) round((yMax -  eigenEnergy)/panelDeltaY) - xAxisRow;
180:
181:                        if(psiFlag) {
182:                                                for(int i =0; i < NUMBER_OF_POINTS ; i++) {
183:                        xScreen4Psi[i] = (int) round( (xForPsi[i] + xMax)/panelDeltaX);
184: yScreen4Psi[i]=(int)round(((yMax*scaleFactor*psi[i])/panelDeltaY)+yOffset);
185:                                                }
186:                         bg.drawPolyline( xScreen4Psi, yScreen4Psi, NUMBER_OF_POINTS-1);
187:                         bg.drawLine(0, xAxisRow+yOffset, panelWidth-1, xAxisRow+yOffset);
188:                                }// if (psiFlag)
189:
190:                        }
191:
192:        void doDrawing( ) {
193:                        repaint();
194:        }
195:        }
```

196:

ไฟล์ที่ 19 : BasePanel.java

```
1:  /* File:BasePanel.java
2:  * Project :Potential Well Simulation
3:  * Author  : Wachara R.
4:  * First Released:  Aug 12,  2010
5:  * Last Updated :  July 03, 2011
6:  */
8:  package classes.SE1D;
9:  import java.awt.*;
10: import java.awt.event.*;
11: import javax.swing.*;
12: import static classes.SE1D.SEConstants.*;
13: import classes.MathTools.Common.Function;
14:  import java.util.*;
15: import java.io.*;
16: /**
17:  * The base panel for all graph demo panels.
18:  */
19: public  class BasePanel extends JPanel
20:
21: {
22:   /** header panel */    private HeaderPanel headerPanel;
23:  /** plot panel  */        private PlotPanel plotPanel;
24:  /** Footer  panel */    private FooterPanel  footerPanel ;
25: /** Left handside panel */             private LeftPanel leftPanel ;
26:
27: /** Potential Function */  protected Function[ ]  functions= new
28:                                 Function[ ] { new SquareWell(2.0, -10.0), new Quadratic(),
29:                                 new  Triangular(), new  TanSquare(), new  InverseCosh() };
30:                 private Function currentFunction;
31:                 int functionChoice = 0;
32:                 double maxXValue = MAX_X_VALUE;
33:                 double minXValue = MIN_X_VALUE;
34:                 int numberOfPoints =NUMBER_OF_POINTS;
35:
36:                 Schrodinger1D se = new  Schrodinger1D(functions[0], minXValue,
37: maxXValue, numberOfPoints);
38:                 double minEn=-10;
39:                 double maxEn = 0;
40:                 double [ ] waveFunction;
41:  // psiFlag  --  true,   if  wave function has been evaluated.
42:  //                        --  fasle  if  user change potentail  function and wave
     Function shoud be recalculated
43:                 boolean psiFlag = false ;
```

```
44:                     double[ ] x;
45:                     int tabIndex=0; // Selected tab in InputPanel
46:                     double guessingEnergy=0;
47:
48:         /**
49:              * Constructor.
50:              * @param graphAttrs the plot properties
51:              * @param xorMode if true, set XOR mode
52:              * @param drawXequalsY if true, draw the X=Y line
53:          */
54:                       protected BasePanel( String headerText)
55:                     {
56:
57:                                 headerPanel = new HeaderPanel(headerText );
58:                                 leftPanel = new LeftPanel(this);
59:                                 footerPanel = new FooterPanel(this);
60:                                 plotPanel = new PlotPanel(this);
61:                 init();
62:             }
63:             /**
64:              * Initialize the graph panel.
65:              */
66:             private void init()    {
67:                                 currentFunction = functions[functionChoice];
68:                 setLayout(new BorderLayout());
69:                 if (headerPanel != null) add(headerPanel, BorderLayout.NORTH);
70:                                 add(plotPanel,     BorderLayout.CENTER);
71:                     if (footerPanel != null) add(footerPanel, BorderLayout.SOUTH);
72:                                 if(leftPanel != null) add(leftPanel, BorderLayout.WEST);
73:             }
74:
75:             protected void setHeaderLabel(String text, Color color)
76:             {
77:                 headerPanel.setLabel(text, color);
78:             }
79:
80:                     void setEigenEnergyLabel(String text, Color color) {
81:                                 footerPanel.setEigenEnergyLabel( text, color);
82:                     }
83:                     protected void setFunction( int choice) {
84:                                 functionChoice= leftPanel.getChoice();
85:                                 currentFunction = functions[functionChoice];
86:         se = new Schrodinger1D(currentFunction,
87:                                 minXValue,maxXValue,numberOfPoints);
88:                                 minEn = se.getMinPotential();
89:                                 maxEn = se.getMaxPotential();
```

```
90:
91:                     }
92:                     public void setPsiFlag (boolean flag) {
93:                             psiFlag = flag;
94:                     }
95:                     public void setMinAndMaxEnergy(double minEn, double maxEn) {
96:                             footerPanel.setMinAndMaxEnergyLabel(minEn, maxEn);
97:     }
98:     protected void  findWaveFunction(double inputEnergy) {
99:                                     int crossing =se.solveWaveFunction(inputEnergy);
100:                                    guessingEnergy = inputEnergy;
101:                                    waveFunction =  se.getWaveFunction();
102:                                    psiFlag = true;
103:                                    x = se.getXCoordinates();
104:    }
105:    protected void findEnergyAtQuantumState(int n) {
106:                                    waveFunction = se.solveEigenEnergy( n);
107:                                    psiFlag = true;
108:                                    x = se.getXCoordinates();
109:    }
110:    void doDrawing ()  {
111:                            plotPanel.doDrawing();
112:    }
113:    int getTabIndex () {
114:                            return footerPanel.getTabIndex();
115:    }
116:    double getMinEnergy() {
117:                            return minEn;
118:                    }
119:     double getMaxEnergy() {
120:                            return maxEn;
121:                    }
122:    public Function getCurrentFunction()
123:                    {
124:                            return currentFunction;
125:                    }
126:
127:    double[ ] getWaveFunction() {
128:                    return waveFunction;
129:    }
130:
131:     double[] getXCoordinates() {
132:                    return x;
133:    }
134:    double getEigenEnergy() {
135:                    return se.getEnergy();
```

```
136:        }
137:        public double getGuessingEnergy() {
138:                        return guessingEnergy;
139:        }
140:
141:        protected double getDeltaX() {
142:                                    return se.getDeltaX();
143:        }
144:
145:        double getMinX() {
146:                                    return  minXValue;
147:        }
148:
149:        double getMaxX() {
150:                                    return maxXValue;
151:        }
152:
153:        double getMaxAmplitude() {
154:                                    return se.getMaxAmplitude();
155:        }
156:        public boolean getPsiFlag() {
157:                                    return psiFlag;
158:        }
159:          //-----------------//
160:           // Event handlers //
161:           //-----------------//
162:           /**
163:            * Mouse clicked event handler.
164:            * (Callback from the plot panel.  Do nothing here.)
165:            * @param ev the mouse event
166:            */
167:           public void mouseClickedOnPlot (MouseEvent ev) {}
168:           /**
169:            * Mouse pressed event handler.
170:            * (Callback from the plot panel.  Do nothing here.)
171:            * @param ev the mouse event
172:            */
173:           public void mousePressedOnPlot (MouseEvent ev) {}
174:
175:           /**
176:            * Mouse released event handler.
177:            * (Callback from the plot panel.  Do nothing here.)
178:            * @param ev the mouse event
179:            */
180:           public void mouseReleasedOnPlot(MouseEvent ev) {}
181:
```

```
182:            /**
183:             * Mouse dragged event handler.
184:             * (Callback from the plot panel.  Do nothing here.)
185:             * @param ev the mouse event
186:             */
187:            public void mouseDraggedOnPlot(MouseEvent ev) {}
188:
189:            /**
190:             * Choose a function.
191:             * (Callback from the function frame.  Do nothing here.)
192:             */
193:            public void chooseFunction(int index) {}
194:
195:            /**
196:             * Notification that the plot bounds changed.
197:             * (Callback from the plot bounds panel.  Do nothing here.)
198:             */
199:            public void conditionChanged() {}
200:          /**
201:             * Process a user input error.
202:             * @param message the error message
203:             */
204:            protected void displayUserError(String message)
205:            {
206:              //  headerPanel.displayError(message);
207:                          JOptionPane.showMessageDialog(null,message
208:                        ,"Error",JOptionPane.WARNING_MESSAGE);
209:            }
210:          }
```

ดรรชนี

T

W