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	$6.02214\times10^{23}\,\mathrm{mol}^{-1}$	
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 $= 1.112650056053618...\times10^{-10}$ 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 $\alpha=\frac{e^2c}{\bar{h}}\times10^{-7}\,N\,A^{-2}$

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

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

ถึงตรงนี้จะเห็นได้ว่า ความเร็วของแสงในหน่วยอะตอม จะมีค่าเป็น $c=\frac{1}{\alpha}\simeq137$

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

ในการวัดความยาว หน่วยอะตอมจะใช้วัดความยาวเป็นรัศมีอะตอมของโบร์ หรือ Bohr radius (a_0) ซึ่งเป็นค่ารัศมีของอิเล็กตรอนวงโคจรแรกของอะตอมไฮโดรเจน โดยที่

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

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

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

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

จะเหลือสั้น ๆ เพียง

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

สามารถนำไปใช้ในการคำนวณเชิงตัวเลขได้สะดวกกว่าแบบเดิมเป็นอย่างมาก

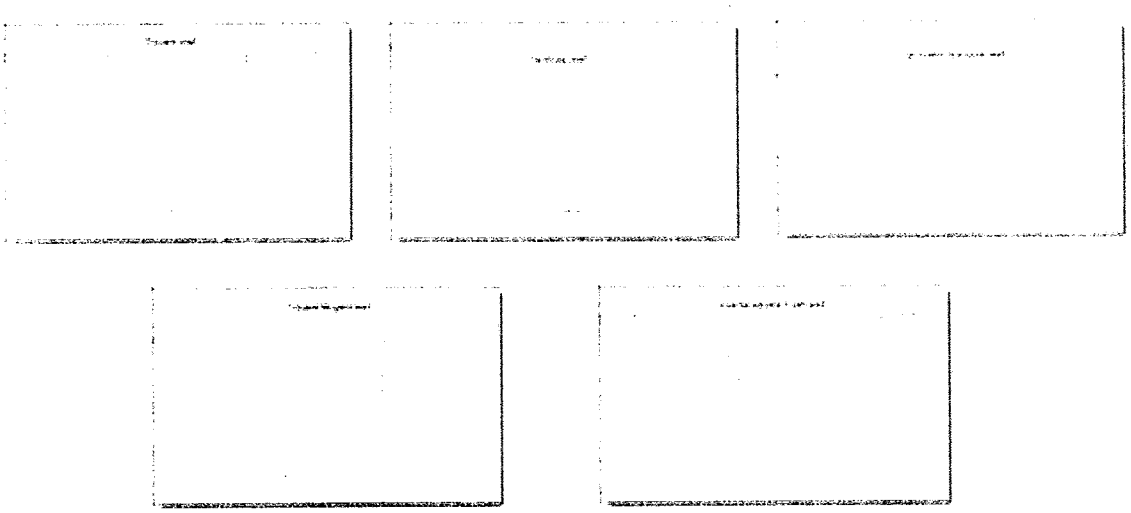
ภาคผนวกที่ 2
คู่มือการใช้งานโปรแกรมจำลองอนุภาคในบ่อศักย์แบบต่าง ๆ

ในการใช้งานโปรแกรมจำลองอนุภาคในบ่อศักย์แบบต่าง ๆ นี้ มีข้อสมมติฐานว่านักศึกษาได้เรียนรู้หรือกำลังเรียนรู้กลศาสตร์ควอนตัมเบื้องต้น ในวิชาฟิสิกส์ 2 หรือวิชากลศาสตร์ควอนตัมสำหรับนักศึกษาสาขาวิชาฟิสิกส์ ถึงหัวข้อสมการชเรอดิงเงอร์ เข้าใจคำว่าฟังก์ชันคลื่น ระดับพลังงานที่สถานะควอนตัมของอนุภาค

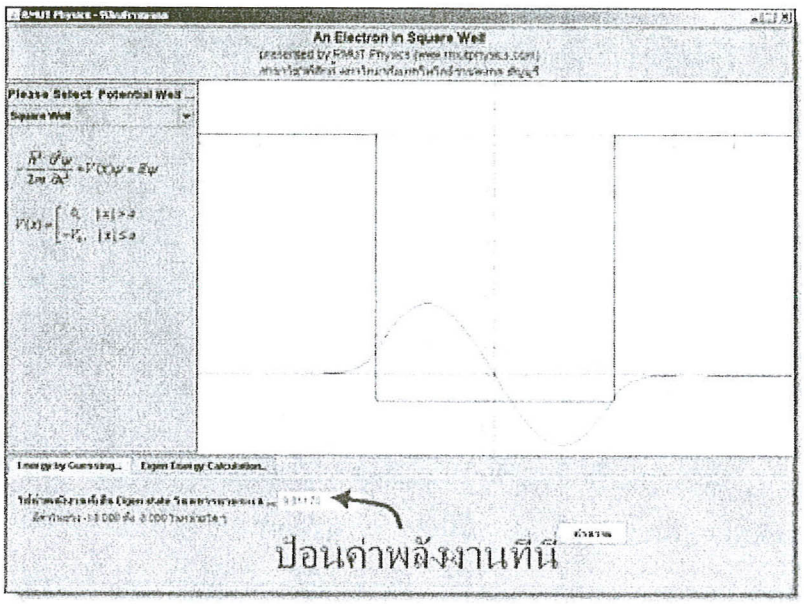
การนำโปรแกรมมาใช้ประกอบกับการเรียน ทำได้ดังนี้

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

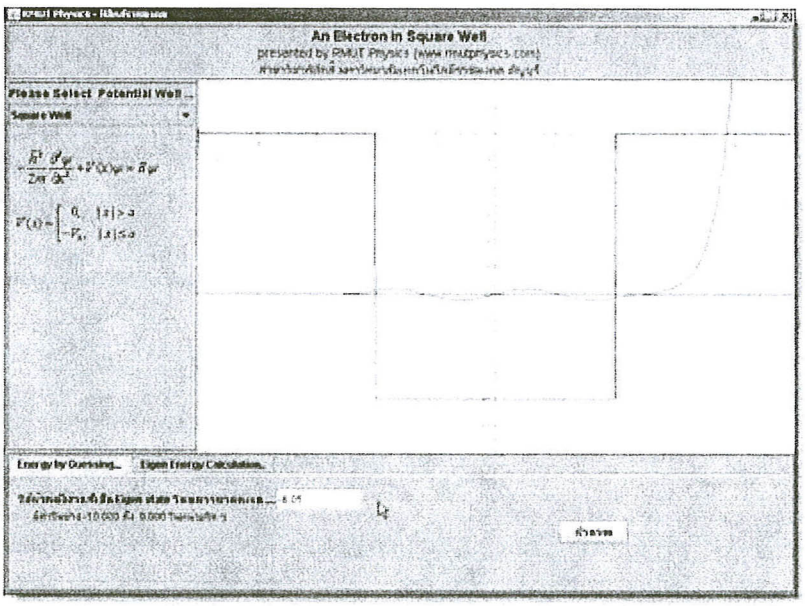
2. เลือกบ่อศักย์ที่ต้องการจะศึกษา มีบ่อศักย์ให้เลือกอยู่ 5 แบบดังนี้ บ่อศักย์แบบสี่เหลี่ยมที่มีความกว้างและความลึกจำกัด บ่อศักย์แบบพาราโบลา(Parabola) บ่อศักย์แบบสามเหลี่ยมสมมาตร บ่อศักย์แบบกำลังสองของค่า tangent และบ่อศักย์แบบส่วนกลับของกำลังสองของ Hyperbolic cosine

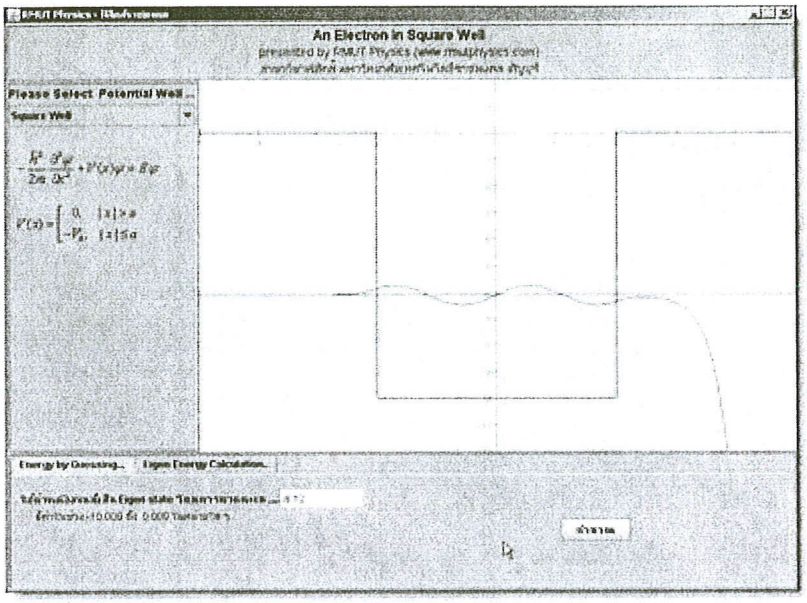


3. เมื่อเลือกบ่อศักย์ที่ต้องการแล้ว จะเห็นรูปกราฟของบ่อศักย์นั้นๆ ปรากฏบนจอด้านขวามือ ให้นักศึกษาทดลองใส่ค่าพลังงานของอนุภาคที่เป็นไปได้ที่อนุภาคยังคงอยู่ในบ่อศักย์นั้น แล้วคลิกที่ปุ่ม “คำนวณ” โปรแกรมจะแสดงฟังก์ชันคลื่นของอนุภาคที่ระดับพลังงานที่นักศึกษาเลือก ถ้าค่าพลังงานที่เลือกเป็นค่าที่เป็นไปได้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



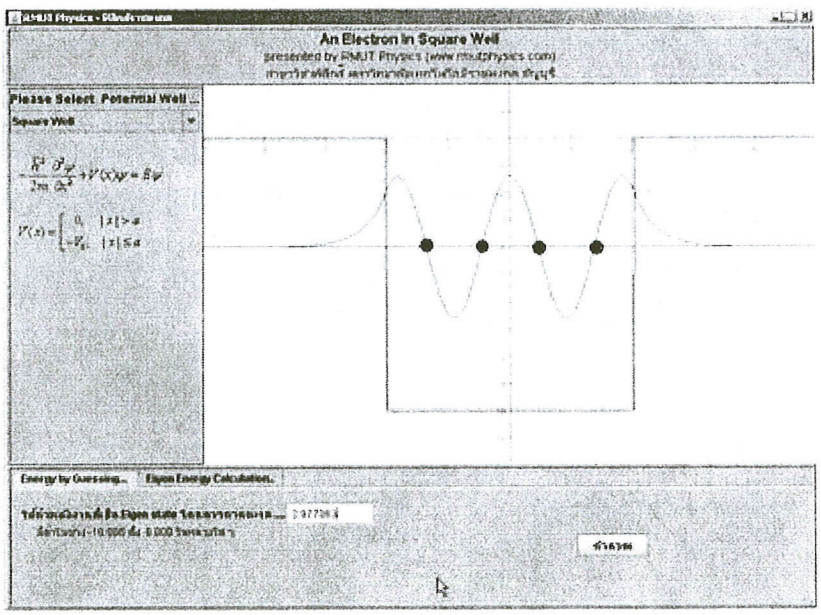
4. ถ้าฟังก์ชันคลื่นที่ได้ มี “หาง” ชี้ขึ้น หรือชี้ลงสู่ค่า infinity แสดงว่าค่าพลังงานที่คาดเดานี้ ยังไม่ใช่ระดับพลังงานที่อนุภาคสามารถมีได้ในบ่อศักย์นี้ ให้นักศึกษาใส่ค่าพลังงานค่าใหม่





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

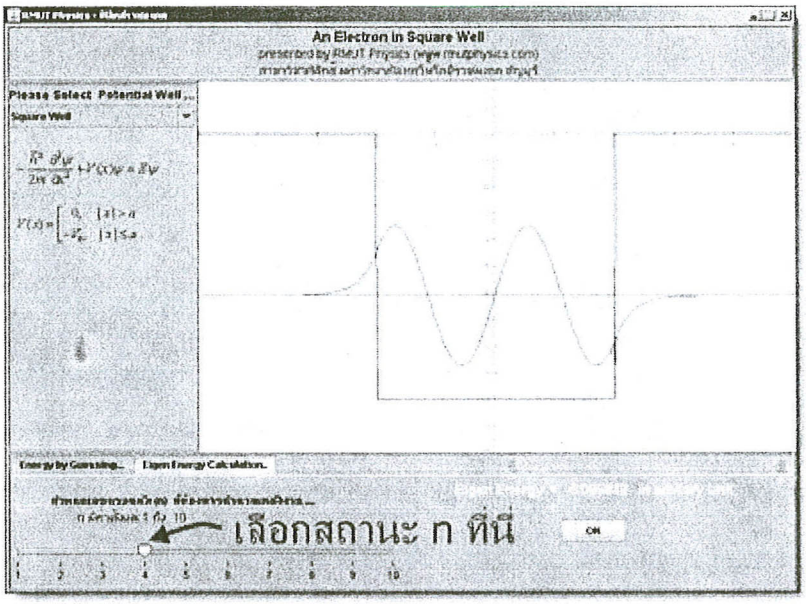
6. เมื่อได้รูปฟังก์ชันคลื่นตามที่ต้องการแล้ว ให้นับจำนวนครั้งที่เส้นกราฟของฟังก์ชันคลื่นตัดกับแกน x สถานะควอนตัม n ที่ระดับพลังงานนั้นหาได้จาก



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

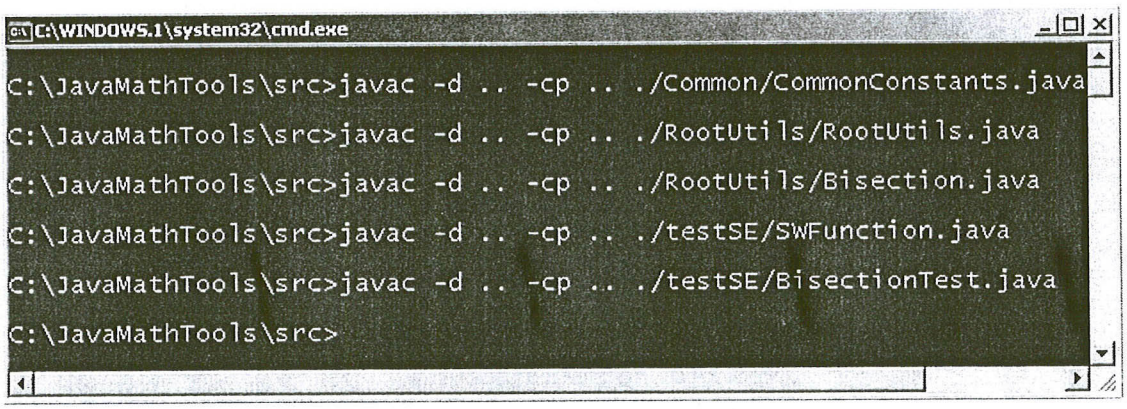
C:\JavaMathTools\src\Javac –d .. –cp .. ./Common/CommonConstants.java

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

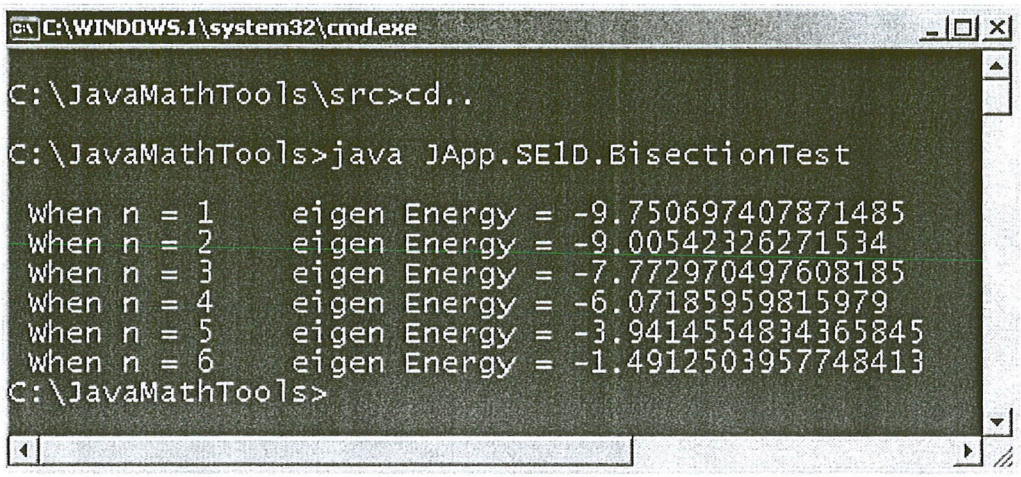
C:\JavaMathTools\src\Javac –d .. –cp .. ./RootUtils/Bisection.java

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

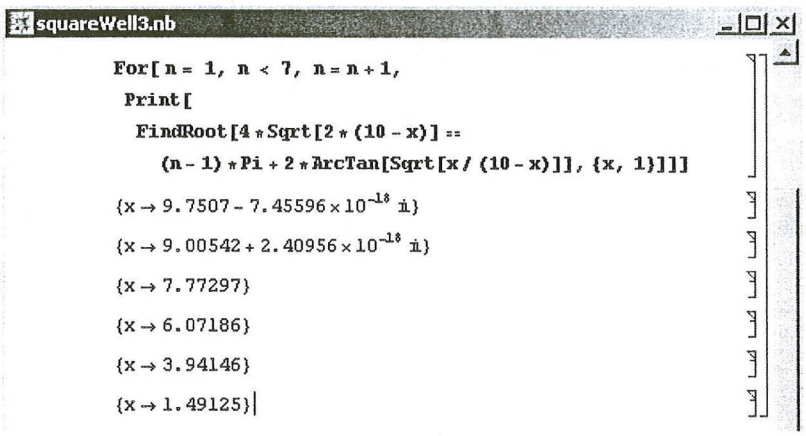
การใช้โปรแกรมสำเร็จรูป Mathematica version 5.1 หารากสมการ

เมื่อเปิดการใช้งานโปรแกรม Mathematica ให้พิมพ์คำสั่งลงไปในหน้าต่าง Notebook ของโปรแกรมดังต่อไปนี้

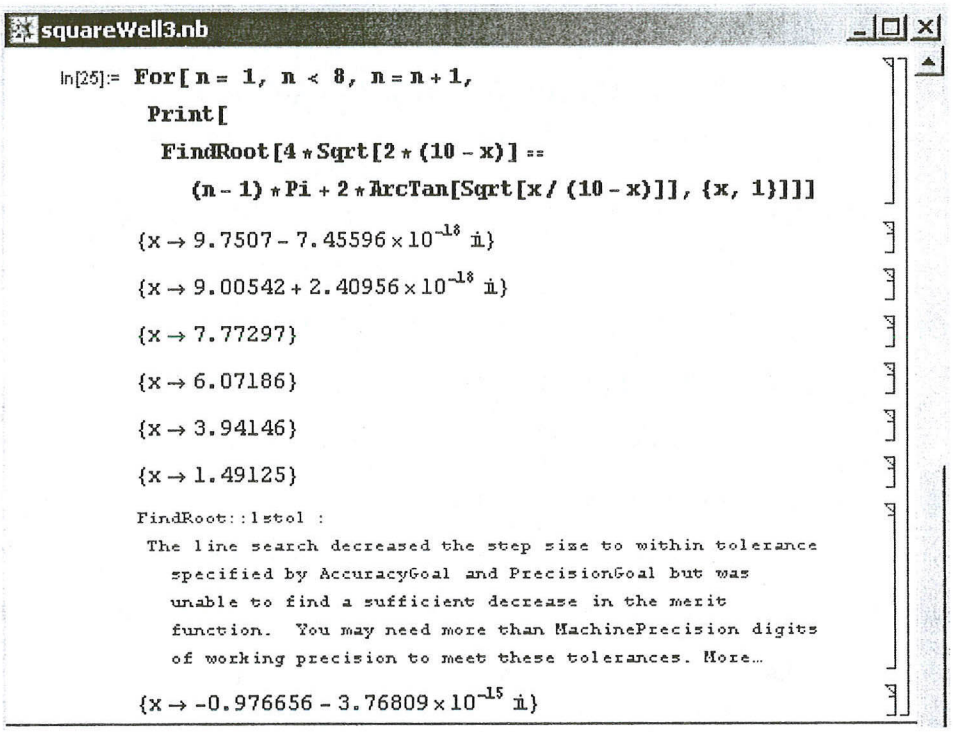
```
For[ n= 1, n < 7, n=n+1, Print[FindRoot[4*Sqrt[2*(10-x)]
        (n-1)*Pi+2*ArcTan[Sqrt[x/(10-x)]],{x,1}]]]
```

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

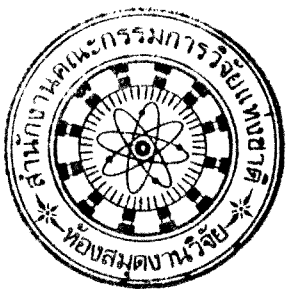
โปรแกรมต้นฉบับ (Source code)

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);
```

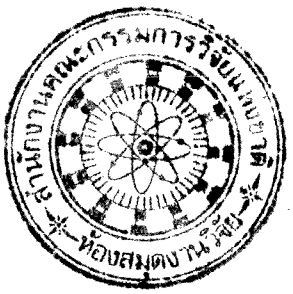
```
78:
79:     LeftPanel.this.basePanel.setFunction(choice);
80:     double                  minEn = LeftPanel.this.basePanel.getMinEnergy();
81:     double      maxEn = LeftPanel.this.basePanel.getMaxEnergy();
82:     LeftPanel.this.basePanel.setMinAndMaxEnergy(minEn, maxEn);
83:     LeftPanel.this.basePanel.setEigenEnergyLabel("Eigen 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

ก

กลศาสตร์ควอนตัม 2, 30, 81

ค

ความเร็วแสงในหน่วยอะตอม 78
ค่าคงที่ของพลังค์ 6
ค่าเจาะจง 7, 43, 53, 81
ค่าไอเกน 20
โคออร์ดิเนต, จุด 62, 63

ง

เงื่อนไขขอบเขต 10, 26, 43, 81

ช

ชเรอดิงเงอร์, สมการ 1,3,5, 23,34, 35, 45, 70, 80

ต

ตัวปฏิบัติการดิฟเฟอเรนเชียล 18
ตัวปฏิบัติการพลังงาน 5
ตัวปฏิบัติการแฮมิลโทเนียน 5

น

นูมารอฟ 24, 34

บ

บ่อศักย์ 3, 41
บ่อศักย์แบบแทนเจนต์กำลังสอง 2, 23, 67, 70
บ่อศักย์แบบพาราโบลา 2, 70
บ่อศักย์แบบสามเหลี่ยมสมมาตร 2, 23, 66, 70
บ่อศักย์แบบไฮเปอร์บอลิกโคไซน์กำลังสอง 2, 23, 68, 70
บ่อศักย์รูปสี่เหลี่ยม 2, 8, 57, 64, 70, 71

พ

พลังงานยึดเหนี่ยวของอิเล็กตรอน 80

ฟ

ฟังก์ชันคลื่น 5, 6, 41, 53, 55, 56
ฟังก์ชันคี่ 16, 17
ฟังก์ชันคู่ 16, 17
ฟังก์ชันเจาะจง 3, 6, 43
ฟังก์ชันอดิศัย 13, 17
ฟูเรียร์, การแปลง 8

ร

รัศมีอะตอมของโบร์ 78, 79

ว

วิธีแบ่งครึ่งช่วง 26, 34, 58, 81
วิธีแบ่งเป็นสี่เหลี่ยมคางหมู 28

ห

หน่วยอะตอม 77

อ

อนุกรมเทเลอร์ 24

อนุพัทธ์, หน่วย 78

ฮ

ฮาร์มอนิก ออสซิลเลเตอร์ 18, 53, 59, 64, 70, 71

A

applet 30

arbitrary unit 77

asymptotic 19

Atomic unit 77

B

BasePanel.java 128

bisection method 26, 34, 58, 81

Bisection.java 87

BisectionTest.java 89

Borh's radius 78

bound state 34

boundary condition 10, 26

C

Command mode 36

CommonConstants.java 85, 97

Coordinate 62, 63

D

derived unit 78

differential operator 18

DOS mode 36

E

Easy java simulation 31

Eigen function 3

eigen value 7, 81

Energy operator 5

EnergyPanel.java 117

even function 16, 17

excel (program) 55

F

Fine structure 78

FooterPanel.java 114

Fourier transformation 8

Function.java 98

G

ground state 7, 80

Guassian wave packet 31

H

Hamiltonian operator 5, 20

harmonic oscillator 18, 53, 55, 59, 64, 70, 71

Hatree 77, 79

HeaderPanel.java 109

I

IIS 33

InputPanel.java 116

InverseCosh.java 109

ionization energy 80

J

jar file 36, 38, 39
Java SDK 33
Java web start 39

K

keystore 39
keytools 39, 40
Kronecker delta function 7

L

LeftPanel.java 111
lower operator 21

M

mainApp.java 94
MathematicaA 33, 58, 81, 92

N

normalize 7, 21, 34 41
NSEApp.java 53
Numarov 24, 34

O

odd function 16, 17
Open source physics 31, 32
orthogonal 7
OSP 31

P

Parabolic well 2, 60, 72, 81
Planck's constant 6, 77
PlotPanel 62, 70
PlotPanel.java 123
Potential well 3
probabilty density 7

Q

Quadratic.java 59, 107
quadrature 28
quadrilaterals 28
Quantization 3
Quantum mechanics 2
QuantumStatePanel.java 120

R

raising operator 21
rest mass of electron 77
rest mass of proton 77
RootUtils.java 86

S

Schrodinger equation 1,3,5, 23, 35
Schrodinger1D.java 45, 98
SEConstants.java 96
self sign certificate 39
SEPanel.java 122
Square hyperbolic cosine well 2, 23, 60, 68, 73, 81
Square tangent well 2, 23, 60, 67, 73, 81

Square well 2, 60, 72, 81
SquareWell.java 57, 106
SWFunction.java 90
SWING 34, 61, 70

T

TanSquare.java 108
Taylor series 24
transcendental function 13, 17, 58, 85
trapezoidal rule 28
Triangular well 2, 23, 60, 73, 81
Triiangular.java 107
tunnel effect 30

W

wave function 5, 6