VPython 在數學教學上的應用

Applications of VPython on Teaching Mathematics

徐愛鳳
中華大學應用數學系
新竹市東香里六鄰五福路二段707號
電話：03-518-6423　傳真：03-518-6521
Email: ahsu@chu.edu.tw

摘要

近年來有相當多研究嘗試著以電腦程式模擬出數學圖形，甚至以互動程式來輔助教學。目的就是利用視覺化的效果來提高學生的興趣及了解度。本文探討運用 VPython 豐富的 3D 介面於數學教育之可行性，另一方面 VPython 的語法簡單易學，在數學課程中加入簡單的 VPython 程式寫作可以幫助完全不懂程式的學生體驗程式寫作的樂趣，尤其可以感受到數學理論與實際應用之間的關係及差距。

關鍵詞：Python，VPython，互動式程式，微積分，泰勒展開式

Abstract

In recent years there were many studies focused on using computer graphic to present mathematics objects, specially geometric objects. Some even use interactive programs to help. The purpose of all these is hoping visualization can provoke students' interest and understanding in mathematics. Our research is in this direction but different from others, we choose a program called VPython which is an extension to the Python programming language. The reason we choose VPython is because it can make 3D graphics display easily and it is easy to learn. In this paper we also present one example.

Keywords: Python,VPython, Calculus, Geometric proof, Taylor series

1. 前言

在一般的數學教育裡，基本的邏輯訓練及定理証明外，就是利用傳統的黑板說明。近年來已有相當多研究嘗試著以電腦程式模擬出數學圖形，甚至以互動程式來輔助教學。目的就是利用視覺化的效果來提高學生的興趣及了解度。應用於數學教學的軟體中較常見的有 Geometric Sketchpad、Matlab、Maple 及 Mathematica 這些數學軟體都見長於數值計算或符號計算，而比較不著重在 3D 視覺化方面的功能。VPython 是建構在程式語言 Python 之上的一個套裝軟體，它在 3D 視覺化方面有豐富的功能及方便的操作介面。

本文將介紹 VPython 軟體及一些利用 VPython 所作的數學例子，並比較它與其他軟體之不同。

2. VPython 的背景

Python 是一種物件導向程式語言(OOP)於 1990 年由荷蘭科學家 Guido van Rossem 所發明。相較於其它語言 Python 的語法結構非常簡單。

例如若要在螢幕上列印一句"Hello World!"

以 C 語言需要寫:

```
#include <stdio.h>
int main( ){
    printf(“Hello World\n”);
    return 0;
}
```

以 C++語言需要寫:

```
#include <iostream.h>
int main( ){
     cout <<“Hello World”<<endl;
    return 0;
}
```

以 Java 語言需要寫:

```
Class HelloWorld {
   Public static void main( String   [ ] args) {
        System.out.println(“Hello World!”);
       }
}
```

而在 Python 只要寫:

```
Print “Hello World!”
```

一行即可。相較於前述語言，Python 顯得語法結構簡單易學。初學者可以透過很短時間的學習即可寫出複雜的程式。VPython 則是一個建立在 Python 上的視覺化圖形模組，其創立者是美國卡內基美隆學生 David Scherer。

3. VPython 的特性

3.1 呼叫方便

VPython 可以讓程式設計者經由簡短的函數呼叫而製造出各種圖形(如直線、長方體、球等)，並可以由攝影機的控制改便其 3D 立體效果。 程式設計者可以經由控制輸出的展示區範圍、並借由移動物體及不同的透視環境，可以很有效地展示出所創造的 3D 模型。

3.2 即時展示

VPython 的應用程式亦可有 run-time “即時”的攝影機控制，滑鼠的左鍵可控制放大、縮小，而右鍵可作 360 度任何方向之旋轉。這對於 3D 物體的展示是很重要的。且將光源設定成固定時，VPython 將自動地計算出物體的陰影。

3.3 物件導向

當我們將圖形快速展示出來加上眼睛視覺暫留現象便形成動畫影片，而程式設計者就像是導演般地創造出精彩的影片。 當然要利用平面螢幕來表現 3D 的效果需要相當複雜的程式，而 VPython 內建指令就是來做這件事情的，甚至於所有物體都是以三維座標的方式放在空間裡並作向量的控制。再者當有些精彩影片或片段程式需要反覆使用時 VPython 也能輕鬆達成，這是因為 VPython 是個物件導向程式語言，當一些 class 被建立後，可借由 import 方式變成模組在未來時可重覆被利用。另外 VPython 是個開放碼，隨時可再接受新的改變。

4. 教學上的運用

VPython 可以用解釋在函數極限的定義，函數可微分的定義，函數積分的定義及函數泰勒展開式等等。本節將探討如何應用 VPython 於微積分教學中函數可微分的定義展示。

4.1 理論部分

給定一函數 $f:(b,c)\to R$ ，在 $a\in(b,c)$ 極限 $\lim_{x\to a}\frac{f(x)-f(a)}{x-a}$ 存在時，稱 f 在 a 點可微分。 嚴格探討此定義必須談到極限定義中的 $\varepsilon-\delta$ 論證(argument)，這對一般學生相當困難，在微積分教學中是一大挑戰，也因此時常被省略不提。然而在互動式視覺化電腦程式中這個可微分定義可以運用 zoom in 及 zoom out 的方法清楚的表現出來。首先畫出曲線

$$\{(x, f(x))\in R^2 \mid x\in[a,a+h]\}$$

然後將兩端點 $(a, f(a))$ 及 $(a+h, f(a+h))$ 的 x 座標差距放大(zoom in)到單位長，也就是放大 $\frac{1}{h}$ 倍。同時整個函數圖形放大 $\frac{1}{h}$ 倍，這時我們便會

看到兩端點 y 座標差距就是

$$\frac{f(a+h)-f(a)}{h}$$ 。

所以如果

$$\lim_{h\to 0}\frac{f(a+h)-f(a)}{h}$$

極限存在，那麼一直放大(zoom in)的結果就是函數圖形漸漸的變成固定斜率的直線，兩者的差距小於肉眼能分辨的 ε 時函數圖形看起來就是這條切線。

學生由此會發現當函數在 $(a, f(a))$ 點可微分時竟然放得越大越分辨不出原函數圖形和切線圖形，反而拉遠(zoom out)了才看出不同，這個與一般越放大應該越清楚的直覺完全不同的有趣現象。

4.2 程式部分

首先我們呼叫 VPython 這個模組(module)：

```
from visual import *
```

接著讓使用者輸入函數式子：

```
f=raw_input("f(x)=")
```

將輸入的式子 f 轉換成可以代入數值的真正函數：

```
F=lambda x:eval(f)
```

下面指令畫出函數圖形：

```
c = curve( x = arange(-1,1+1./20,1/20.) )
c.y = F(c.x)
c.z = 0
```

接著我們用下列程式偵測使用者滑鼠的指令，每按一次滑鼠左鍵，n 的值便會增加 1 而函數圖形就會以滑鼠點下的位置為中心放大兩倍。

```
n=1
while 1:
    if scene.mouse.clicked:
        m = scene.mouse.getclick()
        loc=m.project(normal=(0,0,1))
        x0=loc.x
        c.x=arange(x0-1*(0.5)**n, /
        x0+1*(0.5)**n, (0.5)**n/20.)
        c.y=F(c.x)
        n=n+1
```

如果一開始也呼叫專門負責科學計算的 ScientificPython 中處理函數微分的模組，Scientific.Functions.FirstDerivatives 那麼便可做函數 F 的微分來求得被放大點的切線斜率，並畫出切線來與原曲線比較，程式如下。

呼叫微分的模組：

```
from Scientific.Functions.Derivatives import *
```

然後在定義函數 F(x)之後定義它的一次微分如下：

```
G=lambda x:F(DerivVar(x)) [1]
```

接著在迴圈 while 1 內加上

```
slope=G(x0)
```

並用前述畫出 F(x)函數圖形的方法來畫出切線：

```
tt=curve()
tt.x=arange(-1,2,1.)
tt.y=slope*(tt.x-x0)+F(x0)
```

最後調整視窗大小並將視窗中心移到當初滑鼠按的點：

```
scene.center=p
```

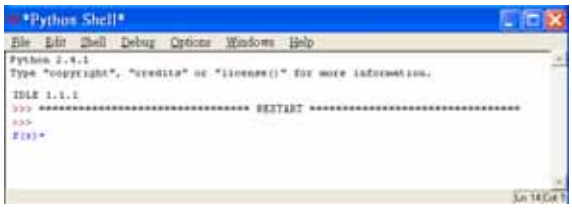
4.3 完整程式

```
from visual import *
from Scientific.Functions.Derivatives import *
f=raw_input("f(x)=")
F=lambda x:eval(f)
G=lambda x:F(DerivVar(x))[1]
scene.background=(0.3,0.3,0.5)
m=20
c = curve( x = arange(-1,1+1./m,1./m) )
c.y = F(c.x)
c.z = 0
tt=curve()
n=1
point=sphere(radius=0.03, color=(1,0,0),visible=0)
while 1:
    if scene.mouse.clicked:
        mo = scene.mouse.getclick()
        loc=mo.project(normal=(0,0,1))
        x0=loc.x
```

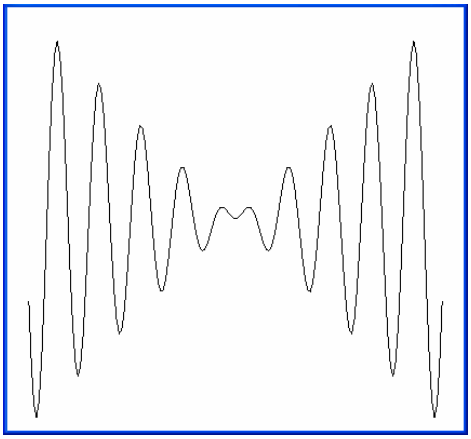
```
p=(x0, F(x0))
slope=G(x0)
point.pos=p
point.radius=0.03*(0.5)**(n-1)
point.visible=1
epsilon=0.5**n
c.x=arange(-1, 1+epsilon/m, epsilon/m)
c.y=F(c.x)
tt.x=arange(-1,2,1.)
tt.y=slope*(tt.x-x0)+F(x0)
tt.color=(1,0,0)
scene.center=p
scene.range=(epsilon,epsilon,epsilon)
n=n+1
```

4.4 實例說明

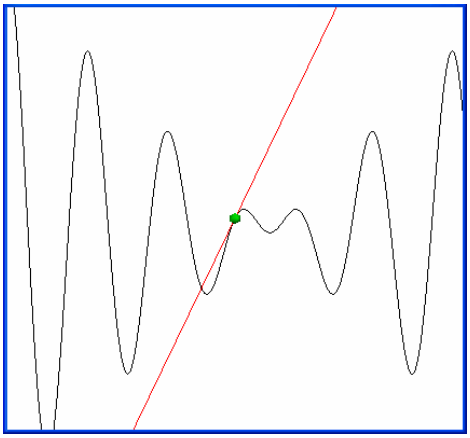
程式一執行便會要求使用者輸入函數



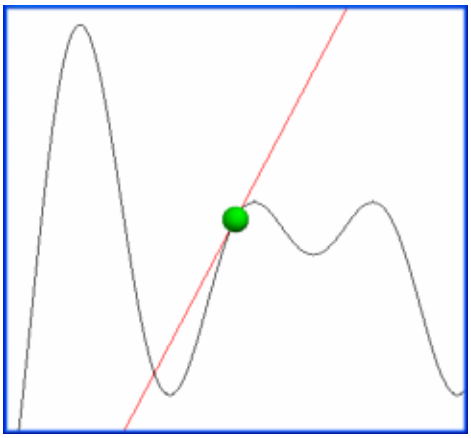
使用者直接輸入函數 $x\cdot\sin(\pi^3 x)$，但在 VPython 語法中 π 用 pi 表示而次方則是**，所以此函數寫成 x*sin(x*pi**3)。 按下輸入鍵後函數圖形的新視窗便會出現。



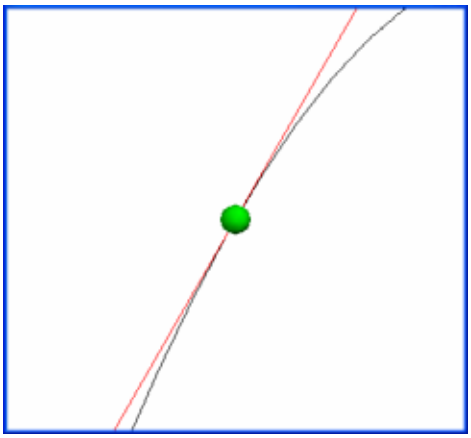
在此視窗曲線上按下一點，便會出現切線。



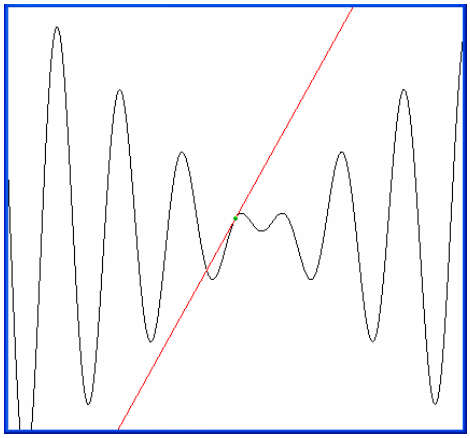
使用者繼續按滑鼠便會看到放大兩倍之後的圖形。



再多按幾次滑鼠圖形就放大到原函數與切線很難分辨的情況。



壓著滑鼠中間鍵往後拉圖形便會放大(zoom out)，這樣原函數與切線又可清楚分辨了。



5. 結論

從上述探討我們看到 VPython 可以用在微積分教學中基本概念的介紹，在實際教學經驗中也看到它的效果。尤其配合 ScientificPython 之後更能在積分理論、高次微分、泰勒展開式等等主題上有更大的發揮。另一方面我們更能經由教導學生瞭解程式本身方面讓學生看到數學與電腦程式之間的關係，尤其 VPython 程式容易改寫，只要稍稍改變一下程式中參數的值或變數之間的關係便能看到不同的結果，學生可因此獲得一些成就感進而願意更深入學習如何寫作程式。

參考文獻

1. D. Scherer, P. Dubois, and B. Sherwood, “VPython: 3D Interactive Science Graphics for Students,” Computing in Science and Engineering, Vol.2, Issue 5, pp.56-62 (2000).