绪　论

计算物理学是利用电子计算机进行数据采集、数值计算和数字仿真来发现和研究物理现象与物理规律的一门现代交叉学科。理论物理学、实验物理学和计算物理学成为物理学的三大分支。早在第二次世界大战期间,美国研究和制造原子核武器的过程中,就采用过计算物理学的方法。当时的情况是：一方面由于原子核材料 U^{235} 的数量有限，不能满足多次试验的需要；另一方面也因为当时世界的政治军事形势十分紧迫,必须抢在德国人之前制造出核武器 还有一方面的原因是描述与核试验相关物理过程的方程组相当复杂，以至用传统的数学物理方法不能获得其解。于是，当时在美国新墨西哥州洛斯阿拉莫斯(Los Alamos) 实验室工作的科学家们不得不动用了数字计算机，这可算作是计算物理学的开端。他们当中有作为领导者的美国加利福尼亚大学理论物理教授奥本海默(J.R.Oppenheimer, 1904～1967) 和著名物理学家康普顿 (A.H.Compton, 1892～1962)。科学家中还有一位现代电子计算机理论的奠定者，即冯诺依曼 (Von Neumann, 1903～1957) 。而且他们获得了成功，于 1945年 7月在美国新墨西哥州南部阿拉莫戈多(Alamogordo)西边的荒漠上成功地爆炸了第一颗铀原子弹。在此后将近半个世纪的时间里，计算物理学主要在原子核物理领域获得应用和发展。 1981 年 3 月,美国哈佛大学 W. H. Press 教授等十一位知名学者向美国国家科学基金会、物理咨询委员会正式提交了发展计算物理学的计划书，这标志着计算物理学业已进入成熟发展阶段。

作为物理学的新型研究手段，计算物理学是理论物理方法和实验物理方法的补充和更新。计算物理学可使理论物理学者从大量烦

需要全本请访问 www.ertongbook.com

琐的计算中解脱出来；从为简化公式而寻求假设的苦思冥想中解放出来。计算物理学可使实验物理中数据的采集和处理实现自动化和高效率,尤其对庞大复杂的实验系统实施控制、协调运作,这绝非手工所能为之。自然界中的物理过程,有些是实验室无法重现的,但却能通过编制程序在计算机中运行来模拟。不仅如此，计算物理学还能对自然界中尚未发现的、假设或预言的物理规律进行仿真,并获得结果。

计算物理学是计算机学、数学和物理学紧密结合的产物,但它毕竟具有物理之个性，而绝非三者的简单混合体。计算物理学总是从物理问题出发,以物理结论为结果,与数值分析的目标不尽相同。例如在物理常微分方程的数值解法中,从数学角度看,高阶的精确的方法总是优于低阶的原始的方法,但从物理的角度看,实际的物理问题未必总存在高阶导数。此时采用高阶方法得不出希望的结果，而采用低阶方法却有明确的物理意义。计算物理学的任务是寻求物理规律,求解物理问题。回忆物理微分方程的建立过程,总是先由物理知识建立一原始的差分关系，然后通过取极限而得到微分方程。数学方法求解微分方程是将微分方程人为地离散化为差分方程，然后由初值递推求解。计算物理学的方法则可直接从原始的差分关系来仿真物理过程。这不仅避免了建立微分方程后又将其离散化的重复劳动，而且还使差分关系中的每一项都保持着直接明了的物理意义。计算物理学方法还特别重视边界处理，因为边界条件是由实际问题决定的，它不但对数值解的精确度乃至对数值计算的稳定性有极大影响，而且对边界条件本身以及解出的结果都具有明确的物理意义。计算物理学的方法还常受到物理问题本身的启示，可利用某些物理现象的直观规律来建立新的计算方法。计算物理学在分析处理大量数据的基础上,非常关注构造和发展近似的解析解,甚至可由此推断出精确的解析解。

物理学与计算机科学的产生和发展有着长久密切的关系。一百多年前，英国发明家巴别奇就已经提出了现代数字电子计算机的原

理和部件。不过他所制造的只是一套复杂的机械系统。 20 世纪初，电真空器件的发展为第一代电子计算机准备了物质条件。 50 年代末，晶体管开始应用于计算机，开创了第二代电子计算机的时代。60年代出现的集成电路，使电子计算机步入了第三代。70 年代出现的大规模集成电路和超大规模集成电路，则使电子计算机随之跨入第四代。历史表明，物理学的进步和成就为计算机科学的发展提供了物质基础。1946 年，第一台数字式电子计算机诞生于美国宾夕法尼亚大学，它的研制者竟是两位物理学者。莫克莱因为研究分子结构需要进行大量计算，而埃克特则是由于有大量实验数据要进行处理。可见,物理学研究中的大量计算和仿真需要,促进了计算机的诞生和发展，同时物理学也为计算机科学的诞生发展准备了人才条件。

第五代计算机 FGCS(the Fifth Generation Computer System)已成为世界各国研究的热门课题。FGCS 将以甚大规模集成技术 VLSI(Very Large Scale Integration)、高速器件、激光技术、语言识别与处理技术、图形处理技术和人工智能等为其技术基础。 FGCS 将是以知识库为后盾,具有智能人机接口的知识信息处理系统。可以展望,随着光计算、光学存储、量子晶体和量子计算机等开拓性研究的深入,新一代计算机将伴随着物理学及相关科学技术的全面发展而来到人间。

第一章　简单物理实验的模拟

简单物理实验的模拟就是通过建立实验模型，编制程序并在计算机中运行，以图形方式和动画形式显示实验过程或给出实验结果。图形程序的编制与显示器和计算机的显示卡有关。下面给出的是常见的三种显示卡的图形显示模式。

1．CGA(Color Graphics Adapter)彩色图形显示卡。CGA彩色图形显示卡的显示模式如下。

	点阵(列×行)	字符(列×行)	字符框(列×行)	颜色(种)
1	320×200	40×25	8×8	4
2	640×200	80×25	8×8	2

2．EGA(Enhanced Graphics Adapter) 增强型图形显示卡。EGA 增强型图形显示卡兼容 CGA 卡的图形显示模式，其它显示模式如下。

	点阵(列×行)	字符(列×行)	字符框(列×行)	颜色(种)
1	320×200	40×25	8×8	16
2	640×200	80×25	8×8	16
3	640×350	80×25	8×14	2
4	640×350	80×25	8×14	16

3．VGA(Video Graphics Array)视频图形阵列。VGA 视频图形阵列兼容 CGA 卡和 EGA 卡的图形显示模式，其标准显示模式如下。

	点阵(列×行)	字符(列×行)	字符框(列×行)	颜色(种)
1	320×200	40×25	8×8	256
2	640×480	80×30	8×16	16
3	640×480	80×30	8×16	2

VGA 兼容卡还有多种非标准图形模式，下列是 TVGA 卡的其它非标准图形模式。

	点阵(列×行)	字符(列×行)	字符框(列×行)	颜色(种)
1	640×400	80×25	8×16	256
2	640×480	80×30	8×16	256
3	800×600	100×40	8×16	256
4	800×600	100×75	8×8	16
5	1 024×768	128×48	8×16	16

下面以 TVGA(1 024×768) 模式为例来说明这些参数与屏幕上像素的位置或字符位置之间的关系。对于点阵而言，水平方向为 1 024个像素,竖直方向是 768 个像素,坐标的原点(0,0)位于屏幕的左上角，右下角的坐标则为(1 023,767)。对字符而言,水平方向可写 128 个字符,竖直方向为 48 个字符，屏幕左上角坐标是(1,1),右下角坐标则为(128,48)。

1.1 简谐振动实验的模拟

由力学知识知道,当物体受到合外力

$$F = -kx$$

的作用,其中 k 为常量,物体的运动方程为

$$\frac{d^2x}{dt^2} + \omega^2 x = 0$$

需要全本请访问 www.ertongbook.com

式中 $\omega^2=\frac{k}{m}$,其中 m 是物体的质量。解微分方程可得

$$A\cos(\omega t+\phi) \qquad (1.1)$$

这里 A 和 ϕ 是积分常量，由初始条件确定。给定一组参数 A、ω 和 ϕ 后,由式 1.1)就可以确定物体相对平衡位置的位移 x 随时间 t 的变化关系。图 1.1 是简谐振动模拟的程序流程图。在图 1.1 中粗实线用于起始与结束框。虚线用于循环框，表示当 $t=0,1,2,\cdots,N$ 时,程序沿虚线箭头方向返回,作循环,当 $t>N$ 时,程序沿实线箭头方向进行。细实线用于工作框，如输入、输出、计算和绘图等操作。t 值的间隔(即程序循环的步长)以及 t 的范围(即 N 的取值)均可自行确定。更新画面就是擦去第 $t-1$ 次循环过程所画的图,再画上第 t 次循环过程要画的图,连续循环下去即可得到动画效果。程序 CP011.C就是在 VGA 显示器上模拟简谐振动的一例。

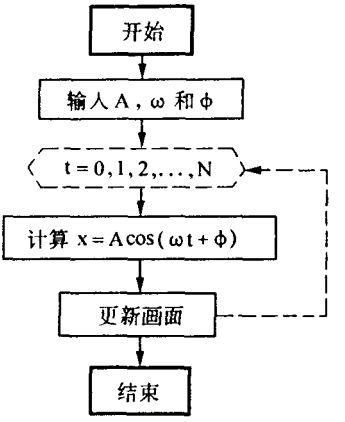

图 1.1 简谐振动模拟程序流程图

```
    /* ----- CP011.C ----- */
#include <graphics.h>
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
main()
{
int graphdrv = VGA;
int graphmode = VGAHI;
int a = 150,t = 0,x = 200,u = 20,N = 1 000,y,r = 4,i;
float w = 0.05,f = 0.8;
```

```
initgraph (&graphdrv,&graphmode," \\ tc \\ bgi");
for (t=0;t< =N;t++)
  {
  for (i=0;i< =100;i++) line(x,u,x,100);
  y=a*cos(w*t+f)+250;
  cleardevice();
  line(x-50,u,x+50,u);
  for (i=0;i< =4;i++)
    {
    line(x+10*i,u,x+10*i+10,u-10);
    line(x-10*i-10,u,x-10*i,u-10);
    }
  line(x,u,x,y);
  line(x+1,u,x+1,y);
  line(x-1,u,x-1,y);
  circle(x,y,r-1);
  circle(x,y,r-2);
  circle(x,y,r);
  }
  while (! kbhit());
  closegraph();
}
```

1.2 振动合成原理的模拟

由力学和振动理论知道，振动方向相同的两个简谐振动

$$x_1 = A_1\cos(\omega_1 t + \phi_1) \tag{1.2}$$

$$x_2 = A_2\cos(\omega_2 t + \phi_2) \tag{1.3}$$

则它们的合振动为

$$x = x_1 + x_2 \tag{1.4}$$

通常,解析式 $x=f(t)$ 的获得是在一定条件下推导出来的。例如，当 $\omega_1=\omega_2=\omega$ 时,可得

$$x=A\cos(\omega t+\phi) \tag{1.5}$$

式中

$$A=\sqrt{A_1^2+A_2^2+2A_1A_2\cos(\phi_1-\phi_2)}$$

$$\phi=\arctan\frac{A_1\sin\phi_1+A_2\sin\phi_2}{A_1\cos\phi_1+A_2\cos\phi_2}$$

或者,当 $A_1=A_2=A$ 和 $\phi_1=\phi_2=\phi$ 时,可推导出

$$x=2A\cos\left(\frac{\omega_1-\omega_2}{2}t\right)\cos\left(\frac{\omega_1+\omega_2}{2}t+\phi\right) \tag{1.6}$$

对于如式(1.2)和式(1.3)所描述的一般情况，要推出类似式(1.5)或式(1.6)那样的解析式并不是一件简单的事情。幸运的是用计算机模拟振动合成原理时，可直接利用式(1.2)、(1.3)和式(1.4)而不必推导其解析解。还可用不同颜色在同一屏幕上同时显示两个分振动及其合振动的振动曲线。程序流程见图 1.2。画 $x-t$ 曲线可以用画点的方法，也可以用画线的方法。画线时，是从上一点画至新点，这样一点接一点地画下去。程序 CP012.C 中采用的是画线的方法。

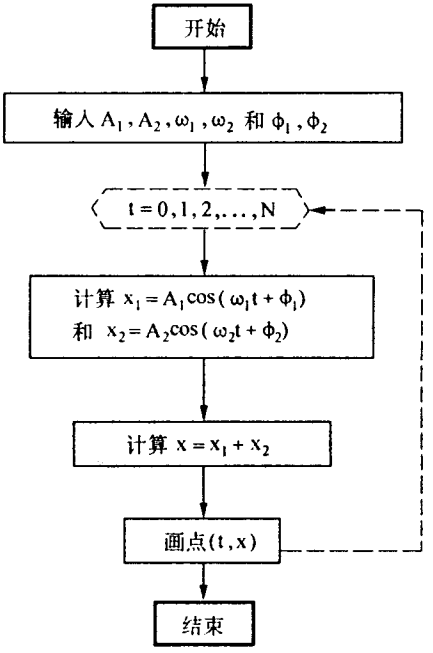

图 1.2　振动合成原理模拟程序流程图

```
    /* ----- CP012.C ----- */
#include <graphics.h>
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
main()
{
int graphdrv = VGA;
int graphmode = VGAHI;
int a1 = 50, a2 = 40, t = 0, N = 1 500;
int y0, y10, y20, y1, y2, y, i;
float w1 = 0.05, w2 = 0.03, f = 0.8;
initgraph (&graphdrv, &graphmode, " \\ tc \\ bgi");
y10 = a1 + 60;
y20 = a2 * cos(f) + 160;
y0 = y10 + y20 + 100;
line(0,60,639,60);
line(0,160,639,160);
line(0,320,639,320);
for (t = 0; t < = N; t + + )
  {
  for (i = 0; i < = 6; i + + ) line(i * 100,0,i * 100,479);
  y1 = a1 * cos(w1 * t) + 60;
  y2 = a2 * cos(w2 * t + f) + 160;
  y = y1 + y2 + 100;
  line(0.5 * t, y10, 0.5 + 0.5 * t, y1);
  line(0.5 * t, y20, 0.5 + 0.5 * t, y2);
  line(0.5 * t, y0, 0.5 + 0.5 * t, y);
```

```
    y10 = y1;
    y20 = y2;
    y0 = y;
    }
while (! kbhit());
closegraph();
}
```

1.3 驻波的模拟

驻波是由频率相同、振幅相同、振动方向平行而传播方向相反的两列波叠加形成的。若取空间两列波位相始终相同的点为坐标原点 $x=0$,并设时间 $t=0$ 时两列波在原点处的位移 $y_1=y_2=A$,则此两列波的波动表达式可以写成

正向波

$$y_1 = A\cos 2\pi\left(\frac{t}{T}-\frac{x}{\lambda}\right) \tag{1.7}$$

反向波

$$y_2 = A\cos 2\pi\left(\frac{t}{T}+\frac{x}{\lambda}\right) \tag{1.8}$$

驻波

$$y = y_1 + y_2 \tag{1.9}$$

给定参数 A、T 和 λ 后，就可用式(1.7)、(1.8) 和式(1.9)来模拟沿正、反方向传播的两列波和驻波。从动态图形中可直观地看出三波之间的关系,并直接验证驻波表达式

$$y = 2A\cos\left(\frac{2\pi}{\lambda}x\right)\cos\left(\frac{2\pi}{T}t\right)$$

所表示出的规律性。驻波模拟的程序流程见图 1.3,其中两个虚线框构成循环的钳套，t 是外循环变量，x 是内循环变量。当循环返回

时先查内循环,只有当 $x > M$ 时才查外循环。若同时又有 $t > N$，则向下沿实线箭头运行，否则哪个变量不满足条件,就返回哪个虚线框,继续循环下一步。下列程序 CP013.C 作为演示驻波的一个例子,可在 VGA显示器上运行。

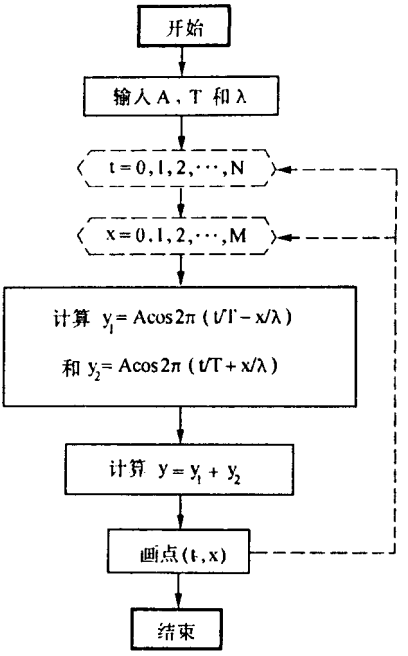

图 1.3　驻波模拟程序流程图

```
    /* --CP013.C-- */
# include <graphics.h>
# include <stdlib.h>
# include <stdio.h>
# include <math.h>
main()
{
int graphdrv = DETECT;
int graphmode;
int a = 50,t = 0,N = 90;
int y0,y10,y20,y1,y2,y,i,x;
float w = 0.2,v = 3.64,f = 0;
initgraph (&graphdrv,&graphmode," \\ tc \\ bgi");
y10 = a * cos(w * (t - 1/v) + f) + 80;
y20 = a * cos(w * (t + 1/v) + f) + 80;
y0 = y10 + y20 + 300;
for (t = 0;t < = N;t + + )
  {
  delay(20);
  cleardevice();
```

需要全本请访问 www.ertongbook.com

```
    for (i=0;i<=3;i++)
      {
      line(i*200,0,i*200,479);
      line(0,80,639,80);
      line(0,300,639,300);
      }
    for (x=1;x<=639;x++)
      {
      y1=a*cos(w*(t-x/v)+f)+80;
      y2=a*cos(w*(t+x/v)+f)+80;
      y=y1+y2+140;
      line(x-1,y10,x,y1);
      line(x-1,y20,x,y2);
      line(x-1,y0, x,y );
      y10=y1;
      y20=y2;
      y0=y;
      }
    }
while (! kbhit());
closegraph();
}
```

为了使动态过程连续逼真，需要时下列改进措施是有效的：

(1)先计算出各点的值，储存在数组中，然后连续画图。这样避免了由于边计算边画图可能造成的画图不连续，给人以一蹦一蹦的感觉。

(2)根据波节的分布，利用其空间分布的规律性可减少大量的重复计算。

1.4 光的多缝衍射的模拟

均匀光源的夫琅和费多缝衍射，在屏上的光强分布为

$$I = I_0\left(\frac{\sin^2 u}{u^2}\right)\left(\frac{\sin^2 Nv}{\sin^2 v}\right)$$

式中，$u = \frac{\pi}{\lambda}a\sin\theta$，$v = \frac{\pi}{\lambda}d\sin\theta$，其中 a 是狭缝宽度，d 是光栅常数，θ 是衍射角，λ 是入射光的波长，N 是狭缝的有效数目。

参数的选择是任意的，但必须服从物理规律。例如缝宽 a 必须大于波长 λ，光栅常数 d 与 a 同数量级，且 $d > a$。这些参数确定后，光强 I 与衍射角 θ 的关系就一一确定了。但还应该注意 θ 值的范围必须考虑计算机允许的值域，以免造成计算机溢出错误。程序 CP014.C 可在同一屏幕上分别绘制多光束干涉、单缝衍射和它们同时作用(即光栅)的光强分布图，其中 a 是缝宽，d 是缝间距，n 是缝数。

```
    /* ----- CP014.C ----- */
#include <graphics.h>
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
main()
{
  float a=0.4,d=0.8;
  int n=4;
  int graphdrv=DETECT;
  int graphmode;
  int A=0,B=120;
  float b,c,r,pi=3.141 593,k,i,f;
```

```
r = B/n/n; k = 2 * pi/1e7; d = d/1 000; a = a/1 000;
initgraph (&graphdrv, &graphmode, " \\ tc \\ bgi");
line(320,0,320,479);
line(0,120,639,120);
line(0,240,639,240);
line(0,479,639,479);
for (f = 1; f < 320; f + + )
   {
   c = pi * d * sin(f/1e5)/k;
   i = r * pow(sin(n * c)/sin(c),2);
   line(320 - A,120 - B,320 - f,120 - i);
   line(320 + A,120 - B,320 + f,120 - i);
   A = f; B = i;
   }
getch();
A = 0;
B = 100;
for (f = 1; f < 320; f + + )
   {
   b = pi * a * sin(f/1e5)/k;
   i = 100 * pow(sin(b)/b,2);
   line(320 - A,240 - B,320 - f,240 - i);
   line(320 + A,240 - B,320 + f,240 - i);
   A = f;
   B = i;
   }
getch();
A = 0;
B = 240;
```

```
    for (f = 1;f < 320;f + + )
        {
        b = pi * a * sin(f/1e5)/k;
        c = pi * d * sin(f/1e5)/k;
        i = 2 * r * pow(sin(b)/b * sin(n * c)/sin(c),2);
        line(320 - A,479 - B,320 - f,479 - i);
        line(320 + A,479 - B,320 + f,479 - i);
        A = f;
        B = i;
        }
    while (! kbhit());
    closegraph();
}
```

1.5 α粒子散射的模拟

设原子核的电荷为 Ze,位置坐标为(x_0, y_0),其质量远大于 α 粒子的质量 m。α 粒子的电荷为 $2e$,其位置用(x, y)表示。

α粒子受斥力为

$$F = 2Ze^2/R^2$$

其中,$R = \sqrt{(x - x_0)^2 + (y - y_0)^2}$。

F 的两分量

$$\begin{cases} F_x = 2Ze^2(x - x_0)/R^3 \\ F_y = 2Ze^2(y - y_0)/R^3 \end{cases}$$

α粒子的加速度为

$$\begin{cases} a_x = \dfrac{2Ze^2}{m} \cdot \dfrac{x - x_0}{R^3} \\ a_y = \dfrac{2Ze^2}{m} \cdot \dfrac{y - y_0}{R^3} \end{cases}$$

α粒子的速度为

$$\begin{cases} v_{2x} = v_{1x} + a_x \Delta t \\ v_{2y} = v_{1y} + a_y \Delta t \end{cases}$$

α粒子的坐标为

$$\begin{cases} x_2 = x_1 + v_{2x} \Delta t \\ y_2 = y_1 + v_{2y} \Delta t \end{cases}$$

为了减轻计算工作量，同时使公式的书写形式更简洁，本节采用的是高斯单位制，或称为绝对静电单位制(CGSE 单位制或 e.s.u.)。另外，运算过程中有两处近似：一是求速度按匀加速运动计算；二是求位置按匀速运动计算，如此递推。如果想要再精确一些，则递推算式就要复杂一些，需要保留的数据也要增加，这里就不详述了。给出一组初始条件,即 α 粒子的初始位置和初速度，即可按上面的式子描绘出 α粒子的运算轨迹。下列程序 CP015.C描绘出一组 α射线被重核散射的图像,其中 $k = 2Ze^2/m$。

```
/* ------ CP015.C ------ */
#include <graphics.h>
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
main()
{
int graphdrv = VGA;
int graphmode = VGAHI;
int i;
float k = 0.75, dt = 0.05, x0, y0, x, y, vx, vy, ax, ay, r;
initgraph (&graphdrv, &graphmode, " \\ tc \\ bgi");
line(400,0,400,480); line(0,240,640,240);
fillellipse(400,240,20,20);
```

```
    settextstyle(1,0,0);
    outtextxy(180,0,"SCATTERED");
    outtextxy(180,25,"ALPHA RAY");
    settextstyle(3,0,0);
    moveto(440,210);outtext("Scattering");
    moveto(460,235);outtext("Nuclear");
    for (i=0;i<5;i++)
      {
      y0=130+25*i; y=y0; x=x0=2; vx=1.5; vy=0;
      r=sqrt(pow(x/10-40,2)+pow(y/10-24,2));
      ax=k*(x/10-40)/pow(r,3);
      ay=k*(y/10-24)/pow(r,3);
      vx=vx+0.5*ax*dt;
      vy=vy+0.5*ay*dt;
loop: x=x+vx*dt;
      y=y+vy*dt;
      r=sqrt(pow(x/10-40,2)+pow(y/10-24,2));
      ax=k*(x/10-40)/pow(r,3);
      ay=k*(y/10-24)/pow(r,3);
      vx=vx+ax*dt;
      vy=vy+ay*dt;
      if (x>639) {goto end;}
      if (x<2){goto end;}
      if (y<2){goto end;}
      line(x0,y0,x,y);
      line(x0,480-y0,x,480-y);
      x0=x;y0=y;
      goto loop;
end:  continue;
```

需要全本请访问 www.ertongbook.com