当代计算机应用丛书

# MC

# 用微机解物理学问题（上）

〔美〕R.克利斯曼著

上海翻译出版公司

《当代计算机应用丛书》

# 用微机解物理学问题(上册)

## 力学和电磁学

(作为哈里德　瑞斯尼克《物理学》的补充)

〔美〕R·克利斯曼 著

沈珊雄　宓子宏 译

上海翻译出版公司

# 内 容 简 介

本书结合哈里德和瑞斯尼克编的《物理学》和《基础物理学》，按通常的物理课程中从力学到电磁学的顺序，讲解了如何将微计算机用于求解物理学问题。其中的程序都以流程形式给出，书末的附录可以帮助读者将流程翻译成实际计算步骤。

对于初学计算机的广大理工科师生来说，这是一本将微计算机应用于物理学的入门书。

J. R. Christman
PHYSICS PROBLEMS FOR
PROGRAMMABLE CALCULATORS
John Wiley & Sons, Inc
1981

用微机解物理学问题(上册)
[美]R·克利斯曼 著
沈珊雄 忽子宏 译
上海翻译出版公司
(上海武定西路 1251 弄 20 号)
新华书店上海发行所发行 苏州师专印刷厂印刷
开本 787×1092 1/16 印张 11.25 字数 275,000
1987 年 6 月第 1 版 1987 年 6 月第 1 次印刷
印数 1—6,000
统一书号：13311·32 定价：2.85 元

# 序　言

在教室和实验室里使用微机*可以为物理学课程增加许多内容。现在，这种工具已不算昂贵，日益普及，我们相信越来越多的教师会在导论或高级物理学课程上采用微机。本书的目的就是以程序流程图及问题的形式，提供可资利用的材料，与微机本身一起，扩充通常的物理学课程。

本书打算为理工科学生通常选修的物理导论课的计算方法提出一些补充。对学生来说，同时修一门微机程序课程是有用的，但并不必需。

本书的结构跟随大多数导论课程中从力学到电磁学的通常的次序。个别章节的引入，是参考了哈里德和瑞斯尼克的《物理学》(第三版)和《基础物理学》(第二版)。这些材料，或者在别的教科书中的类似材料，在着手用微机解题前，应该先被研究过。

不会有超出通常的导论课程的物理学内容。程序说明了各个定律的应用，如牛顿第二定律，库仑定律，毕奥——萨伐尔定律，高斯定律，法拉第定律和克希霍夫定律。许多问题是通过计算来说明这些定律的意义。为了把程序放到上下文中来，于此我们对物理原理作了一些讨论，但这并不意味着可以代替课程教本中的讨论。

在问题中提出的物理内容，比起在通常导论课教本中出现的来说，要更难计算，这对一本包括计算机应用的书来说是适当的。然而，所用到的数学处理，与导论课中为简单问题得到解析解所用的方法常常紧密相关。所有用到的数字计算都作了充分的讨论。

本书可有几种方法使用。在某些情况下，所给程序和较短的练习和问题可作为家庭作业，并在课堂上讨论。许多问题与标准实验室实验相联系，有些就可包括在实验室工作的内容中。较长的问题可作为有兴趣的学生作研究课题。例如在提高班上就可用这些问题。

除课程教科书中的材料以外，并不希望本书中的所有材料都能完成。但在导论课程中加进一些使用微机的内容是可能的。本书提供了各种内容以便选择。我们希望，许多附加问题和研究课题能被选取来适应各种课程结构和时间长短的种种变化。

本书中的程序是为较好的手控程序计算器和较大的计算机设计的，为了做书中的问题，读者必须有接触这种设备的机会。虽然有些程序需要有比小容量手控机器更大的存储容量，但绝大多数问题和练习，不管用何种机器，都能解决。少数问题使用间接寻址记忆储存单元，这对接触具有这一能力的机器来说是有帮助的，但并不必须如此。当做较长的问题时，用能打印结果的机器亦是有帮助的。

大多数问题和练习用手控机器做只化少于半小时时间。有一些问题，最显著的是处理二维运动和画场线图，就要较长时间。对这些问题使用小计算机或大机器是值得的。

程序都以流程给出，使用一种类似于 BASIC 的语言。对大多数微机来说，所给流程需要

---

* 本书内容适用于计算机和带程序的计算器，在译文中都用微机来代。——译者注

翻译成实际计算步骤，流程图中的一行实际上也许代表着许多这种计算步骤。为了对每一个程序给出这些步骤，而且一再把这些步骤交代清楚，这样一旦对四或五种最普通的程序语言中的每一种都这样做，就会使本书极端庞大，且难于使用。反之，流程图倒给出了程序中的要点，而且本书备有附录，可以帮助学生把流程翻译成实际计算步骤。在附录的帮助下，加上对所使用机器的指导手册的研究，学生在做这样的翻译时不会有大困难。

本书给出的程序不一定是最佳的。当然，简单是编写程序的目标。具有编程序经验的学生可以利用特殊机器的能力来写程序，这样可以比这里提出的程序更快更有效。

大多数问题中给出的数据已作了选择，只要机器接受正确的程序并且运转，使得答案可以正确到三位有效数。如果计算结果首先用来画图，精确度可减少到二位，以缩短运算时间。当这样作时，会提到如何修正数据以改善精度。我们打算所有输入数据至少有三位精确，虽然实际上也有给出少于三位数字的。

我们希望学生们在阅读和使用本书时，会感到物理定律的巨大适用范围，并且去实现对这些定律的较高水平上的理解。

对于提供有用的讨论和支持的朋友们，作者向他们表示衷心感谢。

理查·克利斯曼

1980 年 11 月

# 目　录

## 程 序 表

# 第1章 程 序

在这一章，我们要引进整个本书中用来写计算程序的语言。最后所举的例和练习都是矢量分析中的问题。这些可以和哈里德的《物理学》第2章，瑞斯尼克的《基础物理学》第2章，或其它物理教科书中相当的章节相配合。

## 1.1 引言

所有微机都有完成一定的基本函数的能力。它们接受数据（通常通过键盘），储存数据，从存贮器中取出数据，对数据进行算术和其它数字运算，并显示或打印出结果。所有这些都是依照称为程序的一系列指令做的，这些指令储存在微机的记忆系统中，微机就依次执行。

所允许的指令对应于上面提到的基本功能。最重要的数字运算包括加法，减法，乘法，除法，取幂，以及某些有用函数的计算，如正弦和反正弦，余弦和反余弦，正切和反正切，平方和方根，指数和自然对数。有些微机还可以计算其它函数。

机器总是以程序中给定的指令次序来执行的，除非要它不执行。当机器完成一个指令后，它就去进行指令序列中的第二个指令。然而，也有某些指令使机器向前或向后跳到程序中特定的位置，并开始按那里的次序执行指令。这些就称为转移指令，使微机的应用增加许多灵活性。

在第一章的其余部分，我们要讨论本书中如何写指令，并给出足够的程序入门，读者可以循序学习，从而理解后面将提出的程序。

## 1.2 程序语言

本书的目的不是要为任一程序列出键盘排列单，因为这样做会使本书适用范围受到限制。只要程序给定，就容易使用某一种语言为任何一种微机写成键盘序列。

我们使用的语言是类似于 BASIC 的公共程序语言。我们只用描述几乎所有微机能完成运转的那些语言部分，对每一个程序来说，要列出微机去执行程序的键盘是容易的。我们鼓励读者对本章的每一个程序叙述做出具体的执行语言，直到这种翻译适宜运算。几种广泛使用的语言包括在附录中，以帮助读者做这种翻译。本书中给出的每一种程序叙述，都可以在附录中找到对你所用微机适宜的类似陈述。在那里也给出了键盘序列。

现在，我们来描述全书所用的程序语言。

计算器在它存储器中有许多数据存储单元；每个存储单元可以存储一个数。存储的数可以从存储器中调出，在数字运算中使用，再把数字运算的结果存储到存储器中。对大多数计算器来说，各个存储单元由数字标志。我们用符号X1，X2，X3……来代表相应的存储单元1，2，3……。

我们要大大利用这些存储单元符号。它们形成语言的主要成分，出现在几乎所有的程序

叙述中。可以日益明显地看到，大多数时候，这些符号的任何一个可以解释成存储在该单元中的数。例如，可以方便地把X1看做存储在单元1中的数的代表，是不会错的。

物理量通常与一个特殊的存储单元相联系。例如，可以把质点位置矢量的 $y$ 分量总是存储在存储单元1中。然后我们取X1作为该物理量的代数符号。为了书写清楚，理解程序容易，在整个程序中保持物理量和存储单元之间的那种联系是有益的。本书中出现的程序，只要可能，都是这样做的。读者应该尽可能养成习惯，把 $Xn$ 想成存储单元，该单元中的数，和物理量。

四种算术运算写成

$$Xn+Xp$$
$$Xn-Xp$$
$$Xn*Xp$$
$$Xn/Xp$$

上述第一个表示存储在 $Xp$ 中的数加在存储在 $Xn$ 中的数上。其它几个的意义也就不言而喻了。注意：* 这一符号用来表示相乘，以避免与字母 $x$ 相混淆。取幂写成

$$Xn\uparrow Xp$$

意思是：把 $Xn$ 中的数取 $Xp$ 中数的幂。

最广泛用到的函数写成

| | |
|---|---|
| SIN(Xn) | INVSIN(Xn) |
| COS(Xn) | INVCOS(Xn) |
| TAN(Xn) | INVTAN(Xn) |
| EXP(Xn) | LN(Xn) |
| SQRT(Xn) | ABS(Xn) |

大多数这些符号都有确定无疑的解释。INV 用来代表反三角函数。最后一个运算是求出 $Xn$ 中的数的绝对值大小。出于方便考虑，我们把所有角度用弧度表示，用其它单位时，就加以注明。

为了达到本书的目的，包括数字运算的所有指令也为运算结果给出存储单元。这种指令取如下形式：

$$\text{数字运算}\rightarrow Xn$$

这意思解释为：完成箭头左边所表明的数字运算且把结果存储在 $Xn$ 中。

例如：X1 + X2→X3

这条指令告诉机器把X1和X2中的数相加，然后把结果存储到X3中。对于大多数计算器来说，必须先把X1和X2中的数从存储器中调到进行运算的寄存器中，为把计算结果存储起来，通常需要一个特殊的指令。

X1和X2中的数既不会因计算发生变化，也不会因存储指令而改变。如果在执行存储指令之前，X3中有一个数，这个数就会被措掉而用运算的结果代替。

指令可以是比上述指令更复杂的，例如：

$$X1*SIN(X2+X3)\rightarrow X4$$
$$INVTAN(X1*SIN(X2))\rightarrow X3$$
$$SQRT(ABS(X1/X2))\rightarrow X3$$

上述每一条指令并不改变箭头左边提到的存储单元中存储的数。

写指令去改变存储单元的内容，有时是有用的。这种指令是

X1 + X2→X1

在此情形中，X1 和X2 的内容加在一起，得到的和被放入X1 。X1 中原有的内容在这过程中被揩去，很明显，这种类型的指令只有在X1 中原有的数在程序的其余部分不再需要的情况下才被使用。使用这种指令节省了存储单元。在许多情形下，这种指令也使程序更简洁，更容易读。这种指令也保存了存储单元和物理量之间的识别。

为方便计，我们对同一语句中算术运算过程的次序作了假定。取幂首先算出，其次是乘和除，最后为加和减。例如，X1 + X2 * X3 中，在做加法前，乘法先算出来；X1 * X2 ↑ 2 中，在做乘法前，取幂先算出来。在第一个例中，加在X1 上的是X2 和X3 的乘积，在第二个例中，被X1 乘的是X2 的平方。

括号是自由运用的。最内层的括号首先计算，然后是次内层括号，这样依次进行直至最外层括号。在每层括号内，算术运算依照上述次序进行。

前面已经提到，一个数量可以从它存储的存储单元来识别。在我们写数据输入和显示指令

ENTER X$n$

和

DISPLAY X$n$

时，就用到这种概念。第一句指令，机器从键盘接受了一个数，并把它收到的数字存储在 X$n$ 里。第二句指令告诉机器显示存储在 X$n$ 中的数。

从效率的观点说，并不总是愿意立刻存储数字。大多数计算器在输入数字还在显示寄存器时就允许完成在这个数上的运算。尽管如此，本书的指令还是写成输入的数直接进入存储器。这种写法保持了一个数与它的存储单元的识别。

大多数机器对输入和显示的数允许使用科学记法(10的阶次)。需要时，我们也使用这种记法。10 的阶次写在字母E 之后(E 表示指数)。例如 1.075E − 12 表示的意思是 $1.075\times 10^{-12}$。

## 1.3 程序

微机程序包含机器执行的一系列指令（依执行次序排列，除非指定，总是按指令次序执行)。举一些例子足以表明程序该如何建立。这些例子也引出本书中使用的某些惯例。

假定我们希望计算对独立变量 $t$ 各个值的函数 $y=0.3\sin(0.02t^2)$值。我们打算从键盘上输入 $t$ 值。完成这工作的程序为

```
ENTER X1
.3 * SIN(.02 * X1 ↑ 2)→X2
DISPLAY X2
STOP
```

这里，$t$ 值输入并存储在X1，适当的数值运算完成后并把结果代进X2，X2 里的数被显示出来，机器被通知停止了。

这个程序只对 $t$ 的一个值计算了 $y$。为对 $t$ 的许多值计算 $y$，我们要有一个程序，可以回到开始处去接受 $t$ 的一个新值。这就需要用一种称为无条件转移语句或称 GOTO 语句。首先我们把程序的第一句语句标以字母 A，即：

"A"; ENTER X1

然后，在 STOP 之后，我们加上转移语句

GOTO A

当微机接受到这一指令时，它就去寻找标以A的语句，并且接下来便执行这一语句。完整的程序是

```
"A"; ENTER X1
.3 * SIN(.02 * X1 ↑ 2)→X2
DISPLAY X2
STOP
GOTO A
```

指令 STOP 保持着，以便用户可以写下显示的数。当机器重新开始时，它就执行下一个指令 GOTO A。在某些程序语言里，语句不用字母标记，而用数字。

为了给出一个稍为复杂的程序例子，假定我们希望计算等间隔 $\Delta t$ 的一连串 $t$ 值的 $y$ 值。我们令X1为 $t$ 的值，X2为间隔 $\Delta t$，而X3为 $y$ 的值。程序为

```
ENTER X1
ENTER X2
"A"; .3 * SIN(.02 * X1 ↑ 2)→X3
DISPLAY X3
STOP
X1 + X2→X1
GOTO A
```

程序开始时，我们输入 $t$ 的初始值，再输入间隔宽度值。函数就在这个 $t$ 值下计算出来了，显示了结果，机器就停止以允许用户抄录结果。当重新开始时，机器在 $t$ 的现有值(在X1内)上加以间隔宽度 $\Delta t$ (在X2内)。这就产生了 $t$ 的第二个值，且存储进X1中。机器然后回到语句 A，计算 $t$ 新值下的函数值。

包括标记字 A，GOTO 和其间所有语句在内的集合称为一个循环。机器重复地执行循环中的指令序列(当然，除非用户在显示后不再重新启动机器)。当同一个运算必需重复做许多次时，循环是这种程序的重要手段。本书中差不多所有程序都包含循环。

对上述的程序来说，机器每次停止，就显示 $y$ 的一个新值，然后当机器重新开始时，又去计算 $t$ 的下一个值。这样，用户必须注视着机器。如果机器具有打印能力，就可避免这点，可要求机器打印出结果来。当然，这样对用户来说就没有必要去抄录 $y$ 的值了。用 PRINT X3 去代替 DISPLAY 指令，STOP 指令也可删除。

现在，引起了新的问题。机器也许会计算和打印出许多 $y$ 值，超过用户的要求，也就用去大量的纸张。解决办法是使用条件转移或叫 IF 语句。

我们给出一个 IF 语句的例子，它使机器从循环中跳出，假设我们只希望有 $y$ 的前8个值。我们于是让机器对其循环次数计数。我们需要有一个指令，使得循环次数为9时，机器就去执行循环外的指令，而不是再一次去执行循环的语句。这个恰当的语句是

IF X4 = 9; GOTO B

这里X4是机器执行循环的次数，这条指令的意思是：如果X4不是9，机器下一个执行的是IF

后的语句，好象 GOTO B 不在那里一样。反之，如果X4 = 9，则机器下一个执行的指令就是标记B的指令。因为X4 = 9时，任务都完成了，所以语句B是 STOP 语句。

完整的程序是

```
ENTER X1
ENTER X2
0→X4
"A"; X4 + 1→X4
IF X4 = 9; GOTO B
.3 * SIN(.02 * X1 ↑ 2)→X3
PRINT X3
X1 + X2→X1
GOTO A
"B"; STOP
```

要注意，程序中必须表明在循环开始前，让X4 等于 0。这是因为在 A，X4 要加上 1 以计算循环开始的次数。X4 一定要有一个初始值，才可以在这值上加 1，而且初始值一定是 0，才使循环第一次运算时，X4 增值到 1。

还有几种其它形式的条件转移语句。它们是

```
IF X1<X2; GOTO A
```

和

```
IF X1>X2; GOTO A
```

在上述每一种情况中，如果不等式是真，机器就进入标记A的语句。如果不等式是假，则机器继续下去好象转向语句不存在。

## 1.4 流程框图

流程框图是一个图解，对于某一特定的程序来说，流程框图给出了所有的数字运算，而且表明了运算得以完成的次序。它也以 ENTER, DISPLAY, IF 和 STOP 语句表示。1.2 节最后一个程序的流程框图在图 1.1 中示出。

每一句语句放在一个框内：数字指令和 DISPLAY 指令用矩形框，ENTER 指令用缺一只角的矩形框，IF 语句用菱形框，而 STOP 语句用圆形框。

每一个框都用一个数标记，以便讨论程序时可以一个个称呼它们。虽然框的标记数以出现的次序标记，但数字并不是连贯的。缺少的数可以为程序修改时或许会增添的其它语句使用。

连接框的流线说明执行指令的次序。在程序运转时，机器随着箭头从一个指令到另一个指令。在第 130 号 IF 语句，要求机器检验不等式：X4 是否等于 9？如果不等，程序通过菱形框进到第 140 号。如果等式成立，程序就改变方向，进入第 200 号的 STOP 语句。

无条件转移语句并没有明显地在流程框图上表示出来。它只在循环的最后一个语句（第 160 号）中用流线表示出返回到正好第一个语句（第 120 号）前，对程序指明改变方向的流线的出现是一个信号，当流程框图被翻译成机器指令时，必须把转移指令写出来，而且指令必须冠以标记。这里就暗示，在第 160 号后有一个 GOTO 指令，而且第 120 号和 200 号必须标记。

本书的流程框图中用到几个惯例。其中有一个在上一流程框图中已经用到。当相继输入

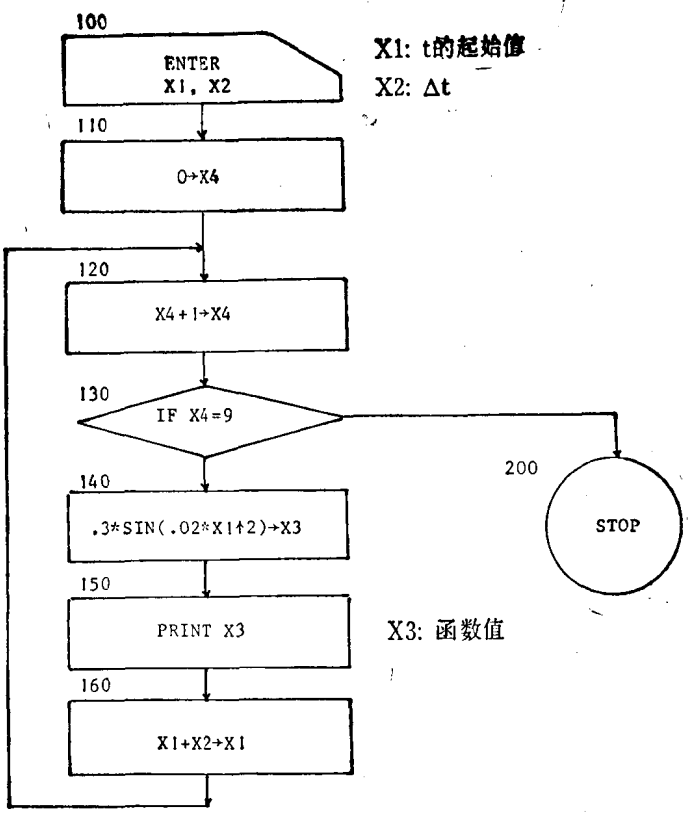


图 1-1 1.2 节最后一个程序的流程框图

几个数时，我们把这些数排列在同一个 ENTER 语句名下，不是每一个数分别写一个语句。操作员要记住，ENTER X1, X2 代表两次分开的"输入"键盘(就象有两个键盘把输入的数送进存储器中)。

对于 DISPLAY 和 PRINT 语句，使用同样的规定。例如，DISPLAY X1,X2 用来代替两个分别对一个数显示的语句。这里也暗示，在每一次分开的显示键盘后，有一个 STOP 指令，以便操作员抄录所显示的数。

本书中有些流程框图，把两个或更多数字运算指令放入同一个框内。同一框内的指令总是以某种方式彼此关联的，把它们归入一框有助于突出相互联系，而且也节省了空间。

对于许多程序，用户必须提供函数。例如，同一程序可以为许多不同的力学问题使用，不同的问题中力作用是不同的，用户要补充程序语句，以便机器能对特殊问题计算力。

例如，假定我们要计算的量是两个独立变量的函数，我们把其中之一存储在X1，另一个存储在X2。我们写出流程框图的一部分：

$$\boxed{f(X1,\ X2)\rightarrow X3}$$

它的意思是：利用X1 和X2 中的数，计算函数，再把结果存储进X3 。在实际使用时，操作员必须补充正确的流程线，以便向机器提供精确的数字运算，完成函数计算。

我们考虑两个矢量 $\boldsymbol{A}$ 和 $\boldsymbol{B}$ 的加法，作为一个程序的另一个例子。如果矢量的直角坐标分量已给出，则和 $\boldsymbol{C}=\boldsymbol{A}+\boldsymbol{B}$ 的直角坐标分量，用

$$C_x = A_x + B_x$$

$$C_y = A_y + B_y$$

和

$$C_z = A_z + B_z \tag{1-1}$$

可以找到。流程框图在图 1-2 中示出。

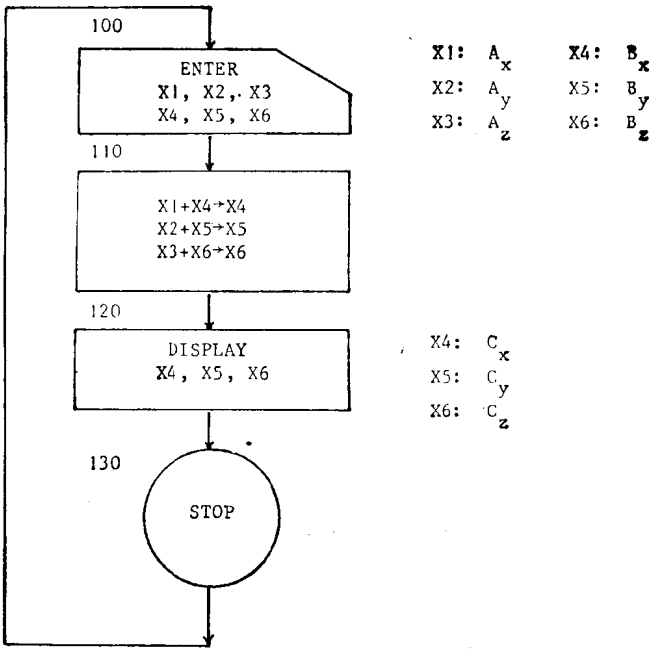


图 1-2 计算两个矢量和的直角坐标分量的程序流程框图

存储单元是

X1 $A_x$，

X2 $A_y$，

X3 $A_z$，

X4 $B_x$（然后 $C_x$），

X5 $B_y$（然后 $C_y$），

和

X6 $B_z$（然后 $C_z$）。

六个分量在第 100 号输入且存储起来。在第 110 号，***C*** 的三个分量被算出，且分别存储入X4，X5 和X6。由于 ***B*** 的分量不再需要，所以这些步骤是恰当的，再说，这样也就节约了机器的存储空间。***C*** 的分量在第 120 号显示。机器停止以便抄录这些数值。当机器重新开始时，它回到了第 100 号，去考虑两个新矢量。这个程序可以作为一个模型，用来计算下面的练习题。

## 1.5 练习

1. *a*. 把图 1-1 的流程框图翻译成你机器适用的键盘序列，并且写上该机器的程序。其中正弦函数的宗量取弧度。

   *b*. 用该程序对 $t$ 从 $t=0$ 到 $t=9$ 的值计算 $y(t)$。每经过一次循环 $t$ 值增加 1。为了让你可以检查你的程序，答案给出如下：

| $t$ | $y(t)$ | $t$ | $y(t)$ |
|---|---|---|---|
| 0 | 0 | 5 | 0.144 |

| | | | |
|---|---|---|---|
| 1 | 0.00600 | 6 | 0.198 |
| 2 | 0.0240 | 7 | 0.249 |
| 3 | 0.0537 | 8 | 0.287 |
| 4 | 0.0944 | 9 | 0.300 |

c. 为你的机器重新编一程序，以计算函数 $y(t)=(t^2+0.6)^{3/2}$，已给出 $t$ 的起始值和间隔宽度。用你的程序求出 $t$ 的值从 $t=0$ 到 $t=9$ 且间隔宽度为 1 的 $y(t)$ 值。

2. 把矢量加法程序流程框图翻译成你机器的键盘序列，编写程序，且用这程序求出下列矢量和。

a. $(5\hat{i}-10\hat{j}+25\hat{k})+(-14\hat{i}+3.7\hat{j}-13.3\hat{k})$

b. $(14\hat{i}+6\hat{k})+(-7.5\hat{j}+3.2\hat{k})$

c. $(\hat{i}+7.1\hat{j})+(3\hat{i}-17.4\hat{k})$

3. 设计一个程序的流程框图，以计算给定直角坐标分量的两个矢量的标积。当机器完成计算时，回到开始处，考虑两个新矢量。把这流程框图翻译成你机器的键盘序列，把它输入机器，且用这程序计算下列标积。

a. $(3\hat{i}+4\hat{j}+3\hat{k})\cdot(\hat{i}+3\hat{j}+2\hat{k})$

b. $(3\hat{i})\cdot(7\hat{i}-3\hat{j})$

c. $(5.7\hat{i}-6.2\hat{j}+2.1\hat{k})\cdot(2.3\hat{i}+7.6\hat{j}+3.8\hat{k})$

4. 矢量的大小可取矢量与它本身的标积的开方根来得出。为一程序设计流程框图，以计算已给定直角坐标分量的矢量的大小。当机器完成计算时回到开始端以考虑新矢量。把这流程框图翻译成你机器的键盘序列，把它输入机器且用此程序计算下列矢量的大小。

a. $7.1\hat{i}+13.2\hat{j}-16.1\hat{k}$

b. $12.6\hat{i}-7.1\hat{j}$

c. $-5.8\hat{j}+4.1\hat{k}$

5. 两矢量间的夹角可以用标积除以两矢量大小的乘积后，再求反余弦函数值得到。例如 $\boldsymbol{A}$ 和 $\boldsymbol{B}$ 之间的夹角 $\theta$ 由

$$\theta=\cos^{-1}\frac{\boldsymbol{A}\cdot\boldsymbol{B}}{AB}$$

给出。设计一程序的流程框图，求给出直角坐标分量的两矢量之间的夹角。当机器完成计算时，机器程序回到开始端以考虑两个新矢量。

同一个余弦存在着不止一个角度。一般要求正的且小于 $\pi$ 弧度的角。由于大多数机器自动地给出这个角度，所以没有必要特别注意。

把流程框图翻译成你机器的键盘序列，把它输入机器，并用此程序求出下列各对矢量之间的夹角。

a. $(7.4\hat{i}+3.7\hat{j})$，$(3.9\hat{i}+4.7\hat{j})$

b. $(-3.7\hat{i}+1.2\hat{j})$，$(3.7\hat{i}+1.2\hat{j})$

c. $(4.2\hat{i}+12.1\hat{j}+7.3\hat{k})$，$(2.3\hat{i}-7.8\hat{j}-5.3\hat{k})$

6. 练习题 5 中的程序可用来找出一个矢量与一个直角坐标轴所成的角度。这时一对矢量中的第二个矢量就是沿着所选坐标轴的任何矢量。试利用练习题 5 中的程序求下列夹角。

a. $3\hat{i}-7\hat{j}$ 和 $x$ 轴之间的夹角

b. $3\hat{i}-2\hat{j}+6\hat{k}$ 和 $x$ 轴之间的夹角