BASIC 土力学

G.W.E.Milligan G.T.Houlsby著

张喜发 译 张运中 校

长春地质学院《世界地质》编辑部

ν	泊松比
A，B	孔隙压力系数
C_c，C_s	压缩指数和回弹指数
K_o，K_a，K_p	土压力系数
C_v	固结系数
U_t	固结度
T_v	固结时间因数
m_v	体积变化系数
F	安全系数
N_s	稳定数
m，n	稳定系数
r_u	孔隙压力比
N_c，N_q，N_γ	承载力系数
i_c，i_q，i_γ	承载力倾斜系数

注释：

1．应力分析中本文一律采用下列脚标：

x，y，z 坐标方向

1，2，3 主值

2．除非另有说明，本书使用的其它脚标皆表示如下意义：

d	干土的
sat	饱和土的
u	不排水的
h	水平方向的
v	竖向的
a	主动的
p	被动的
o	初值或表面值
f	破坏值

3．有关力和图形特征的符号另在正文或图表中详细说明。

中 文 译 序

本书根据牛津大学G.W.E.Milligan和G.T.Houlsby合著的巴特沃思BASIC丛书之一——BASIC土力学第1版（1984）英文本译出。

这是一本按崭新形式写成的土力学参考书。书中作者对实用土力学的主要问题和用BASIC语言设计计算机程序的方法，作了非常清晰简洁的介绍。

随着土木建设事业的发展，土力学在土木工程、岩土工程中的重要地位有增无减，越来越多的人要求掌握土力学的知识，尤其是实用土力学的主要知识。可以认为，这本书用来满足这种要求是容易收效的。它篇幅不大，但是主要的概念和基本理论，都已精辟地作了概括。书中还包含了一些比较新的成果和内容。例如临界物态概念、剑桥模型等，这在一般土力学教科书和参考书中还很少见有介绍。

书中的程序设计实例是本书的重要内容。这些实例覆盖了大量土力学的基本问题。很多程序可供直接利用。但是作者更大的用意是希望能够帮助读者掌握用BASIC语言编制程序的能力。因此所举实例并不完全着眼于解决个别具体问题，而着重在介绍各种方法的运用。

书中还列出了一些习题。这些习题显然也是作者精心选出的。难怪作者极有信心地称，只要读者细心地学习了书中的程序，并完成了所列的习题，就一定能够有把握地运用BASIC程序来解决土力学中的实际问题。

近年来，我国各工科院校和生产单位微型计算机的配置日渐普及。专业人员利用微机的机会越来越多。我们现在把这个中译本奉献给读者，希望能对有关专业工作者和在校师生有所裨益。

本书的翻译得到了谭周地、唐大雄、肖淑芳等同志的支持，对此深表谢意。

由于水平有限，翻译中不当之处在所难免，恳请批评指正。

译者

1988年1月

序　　言

近年来，计算机与计算机程序设计的意义已经迅速地引起了人们的重视。特别是微机问世以来，人们的这种认识更加增长了，认识到它们重要性的人数也越来越增多了。现在，大多数工程技术工作者都有广泛的机会利用各种各样的计算机软、硬件。可以说，人们已开始把使用计算机当成一种很平常的事情。

现今在英国，计算技术已普遍列入各工科高等院校的课程目录，甚至有些中等学校也开设了这门课程。但是，由于讲授这门课的多为计算机专家或数学家，他们一般难于做到把这门课同学生所攻读的工程专业课联系起来，因此不少学生常觉得这门课不好学。可是，如果把工程学同计算技术联系起来教学，那就有利多了。工程学知识能帮助我们学习计算技术，因为工程学使我们能看清问题的本质，通过许多实例使我们又有可能学习、练习和研究计算机程序设计的方法。由此可以看出，掌握计算技术本来就该是工程师必须具有的一项基本能力。计算技术也能帮助研究工程学问题，因为只有能够运用计算知识清晰地描述问题中的公式和通晓有关处理方法，才能编写出一个成功的研究工程问题的计算机程序来。也正因为如此，这种把二者联系起来的教学方法最能引起学习者的兴趣与热情。而这不仅对于刚刚接触实际的技术工作者来说很有价值，就是对于那些已经从事专业工作、但不曾学过计算课程或曾碰到过某些计算难题而穷于应付的工程师们来说，也是十分有意义的。

本书所用计算机程序设计语言为BASIC语言。这种语言在计算机专家看来不是很好的，因为它不具备结构化性质。但是它可以普遍地用作微机内部语言。除此之外，对工程师们来说，还有其它优越之处，这种语言使用简单英语语句，因而较易学习和记忆，工程师们初学即无问题，即使以后不常用也不会生疏。用这种语言设计程序可以迅速完成，因为它无须再经翻译、连接和编辑等例行程序，这对初学者，对老行家来说，都是方便的。进一步说，由于写出一个程序一般只需几个小时，所以学习者也不致于轻易失去学习它们的兴趣与信心。

本书是按BASIC语言应用于土力学这门重要工程学科的方式编写的，旨在帮助有关专业读者掌握BASIC语言的程序设计方法。书中列举了大量短程序，阅后当能相信学习计算技术不仅是必要的，也是不难的。对已从事实际工作的工程师来说，许多程序可以直接使用，但是作者更主要的是希望通过所举实例说明各种程序设计方法，从而使技术工作者能够掌握和运用这些方法去解决各自的具体问题。

本书第一章为BASIC语言简介。第二章介绍了土力学基础和土的工程性质。第三章至第七章包括了土力学基本教程通常介绍的几个题目：土的鉴别、分类和试验；土压力；边坡稳定性；地基等。土中水的流动对岩土工程的理论和实践都是一个重要方面，因而大多数土力学书籍均单独辟为一章予以介绍。然而，由于这方面的问题在本套丛书

已出版的《BASIC水力学》（P.D.Smith）一书中的渗流一章作过详细介绍，为避免重复，本书不再赘述。

各章均包括有关基本理论的简述、计算机程序设计实例和一套习题等三部分内容。理论部分显然不是对所研究问题的全面介绍，只不过是一个大纲或摘要。程序设计举例按实际问题提出，并给出一个合理的程序及其输出和若干条程序注释。其中的程序注释主要注明程序的结构及程序究竟应用哪些理论，哪些公式。习题包括要求修改或扩展所举例中的程序，以及编写全新的程序。读者通过学习书中实例，并完成所给习题，定能掌握编写BASIC程序的方法，也能学到有关土力学方面的知识。

利用计算机把计算结果以图形的形式输出，或者在屏幕上显示出来，或者打印在纸上，也许会使许多程序更有用。可惜的是，编写计算机绘图程序的方法往往因计算机系统的不同而各异，因此本书未予列举这方面的例子。然而希望读者尽量利用所使用的计算机的绘图功能，在这方面付出一点努力是非常值得的。

在此谨向M.J.Iremonger和P.D.Smith两位编辑表示致意，感谢他们在编辑出版本书中给予作者的帮助，特别是Smith先生，他首先鼓励作者编写此书。同时也感谢Sue Clarke和Sally Schofield为本书打印手稿，感谢Judith Takacs为本书准备插图。

G.W.E.Milligan

G.T.Houlsby

主 要 符 号

e 孔隙比

n 孔隙率

v 比容

w 含水量

S_r 饱和度

G_s 土颗粒比重

A 气体含量

γ 容重（单位体积的重量）

ρ 密度（单位体积的质量）

σ, σ' 法向总应力和法向有效应力 } 见注释 1

τ 剪应力

u 孔隙压力

$p=\frac{1}{3}(\sigma_1+\sigma_2+\sigma_3)$ 平均法向应力

$q=(\sigma_1-\sigma_3)$

$s=\frac{1}{2}(\sigma_1+\sigma_3)$ } 平面应变条件

$t=\frac{1}{2}(\sigma_1-\sigma_3)$

ε 直接应变（指拉伸或压缩时的线应变——译注）

γ 剪应变

$\varepsilon_v=(\varepsilon_1+\varepsilon_2+\varepsilon_3)$ 体积应变

$\varepsilon_s=\frac{2}{3}(\varepsilon_1-\varepsilon_3)$（三轴试验），$(\varepsilon_1-\varepsilon_3)$（平面应变）

M 土的临界物态摩擦常数

ϕ 内摩擦角

c 内聚力

c_u, c_a 不排水抗剪强度，粘着力

δ 墙的摩擦角

E 杨氏模量（弹性模量）

G 剪切模量

目　　录

第一章 BASIC 语言简介

1.1 BASIC 程序设计语言

本书所用的程序一律按BASIC程序设计语言书写。BASIC是“Beginner's All-purpose Symbolic Instruction Code”（初学者通用符号指令代码）一词的缩写，是美国Dartmouth学院研究出来的一种简单的计算机通用语言。这种语言是为在分时计算机系统上使用而设计的，但当把它用到微型计算机上以后，便很快地广泛普及了。BASIC语言易学易用，利用它可以对一个程序迅速地完成书写，在计算机上，让它运行。如果出现错误，易于修改，让它重新运行。基本BASIC语言的主要缺点是不具备结构化性质（见1.4），但对于较短的程序（如后面各章中的程序）来说，这显然不是很大的问题。

编写本书的目的在于，通过把BASIC应用到某些有关工程问题上来帮助读者掌握这种语言的应用。相信通过学习书中的实例，并尽可能地做些模仿练习，然后再试着解决一些具体问题，这一目的是能够实现的。本书将不专门讲授BASIC语法，只在下节对所用的基本BASIC语言简要地予以介绍。

1.2 BASIC语言的基本组成

1.2.1 数学表达式

本书中列举了若干程序实例，列举它们的主要目的之一在于计算土力学中出现的方程式。这些方程式包含数字常数（如摩擦角ϕ'）、变量（如x）和函数（如sine）。数字常数可以是整数（如36），也可以是非整数的其它实数（如36.1）。对于一些很大的或很小的数字（如3.61E6等于3.61×10^6）则可用指数形式表示。数字变量用一个字母、一个字母后跟另一个字母或一个字母后跟一个阿拉伯数字表示（如E，EA或E1）。运算，例如平方根，可使用内部函数(如SQR(X))来完成。括号内的自变量（X）可以是一个数字、一个变量或一个数学表达式。对三角函数（SIN(X)、COS(X)等），自变量总是被理解成是以弧度来度量的。其它的函数还包括自然对数和它的逆（分别为LOG和EXP）、取自变量绝对值的ABS和取自变量整数部分的取整函数（INT）。

数学方程式还包括一些运算符如加和减等。这些运算符有一个分层结构，按这个结构计算机对一些运算符先予执行，而对另一些则在后执行。按这个结构，这些运算符的递降顺序是、

乘方（∧）

乘（*）和除（/）

加（+）和减（-）

例如，乘法在加法前执行。如果运算符处在同一层次上，计算机便从左到右运算。括号表示超前任何一个运算符。因此$-\frac{a+b}{3c}$可写成(A+B)/(3*C)或(A+B)/3/C。但是建议不要用这后一种表达方式，因为它容易含糊不清。

在某些计算机中，变量名只限于用一个字母，但通常允许采用由两个或两个以上的字母组成的变量名。为了使变量名比较便于记忆，本书许多程序中使用两个字母的变量名。

1.2.2 赋值语句

BASIC程序是由一系列的语句组成的，这些语句规定着计算机的工作步骤。运行中，计算机先后把一些数值分配给每一个变量，其中有的是由键盘输入到程序中的数据规定的，有的则是程序运行中生成的，例如经由赋值语句赋给的赋值语句（LET语句）的形式为

语句标号①〔LET〕变量=数学表达式

其中LET通常不是必须的，因此可以省去。（今后，凡表达式中所有非必须部分都将用方括号括起来，表达成这样：〔非必须的〕）。例如，一元二次方程式的某一个根

$$x_1=\frac{-b+\sqrt{(b^2-4ac)}}{2a}$$

可表达成这样的语句：

```
100 X1=(-B+SQR(B^2-4*A*C))/(2*A)
```

重要的是需要认识到赋值语句本身并不是一个方程式，它只是把等号右边的表达式的数值赋给等号左边的变量的一个指令。因此允许有这样的语句

```
50 X=X+1
```

其意义是X值增加1后再赋给X。

每一变量在任何一次计算时只能有一个值，除非它是下标变量（见1.2.7）。

应注意，所有BASIC语句（即所有程序行）都要加标号，标号规定了执行语句的顺序。

1.2.3 键盘输入语句

就“人-机对话式”程序来说，程序员可以在程序运行中，用键盘把数值输入给相应的变量。键盘输入语句（INPUT语句）的形式为：

语句标号 INPUT 变量1〔，变量2，…〕

例如

```
20 INPUT A,B,C
```

在程序运行中，当执行到这一语句时，计算机屏幕就显示出“？”，等待程序员为这

①语句标号即行的标号——译注

些变量赋值，例如

? 5,10,15

后面的意思便是使上边程序中的变量A＝5，B=10和C=15。

这一节乃至本书中的所有其余部分，必须由程序员打入（而不是由计算机打入）的输入都通过在下边划线加以表示。程序员当然不必在输入下边划线，而且在正常使用计算机时，也不会出现这种划在数字底下的横线。

如果数据数量很大，或者还希望把这些数据保留在机器里，那么可以采用另一种输入方式。这就是采用一个读入语句和一个或若干个数据语句，读入语句(READ语句)①的形式为：

语句标号 READ 变量1〔，变量2，…〕

例如

20 READ A,B,C

数据语句（DATA语句）②的形式为

语句标号 DATA 数字1〔，数字2，…〕

例如

1 DATA 5,10,15

或

1 DATA 5
2 DATA 10
3 DATA 15

DATA语句可放在程序中的任何位置，而把它们放在程序开头或末尾更为方便。这样变更起来也比较容易。

使用内储数据时，在一次运行过程中，有时需要不止一次地从数据起点读数据。这可以用以下形式的恢复数据区语句（RESTORE语句）来实现：

语句标号 RESTORE

1.2.4 输出语句

数据和计算结果等的输出使用显示语句（PRINT语句）③执行，形式是

语句标号 PRINT清单

输出清单可以是变量或表达式，例如

200 PRINT A,B,C,A*B/C

或用括号括上的文字，例如

10 PRINT"INPUT A,B,C,IN MM"

或既有文字又有变量，例如

---

①或称读数语句
②或称置数语句
③或称打印语句
} ——译注

```
300 PRINT "STRESSIS",S,"KN/M∧2"
```

输出清单中的各项要用逗号或分号隔开。用逗号时，将以标准格式输出，每列约占15个格。用分号时，则将以紧凑格式输出，如果分号在清单末尾，则不换行。如果清单是空的，则显示出一个空行。

应当注意，下面两种情形都需同时使用PRINT语句：一是使用“运行时”输入的时候，需要适时了解该输入什么；二是使用READ/DATA语句的时候。第二种情形下，如未用PRINT语句，程序员将看不到数据记录。

1.2.5 条件语句

经常有必要使程序当且仅当某种条件被满足时能采取某种行动。这可用条件语句（IF—THEN语句）加以执行，其形式是

语句标号 IF表达式1｛条件符｝表达式2 THEN GOTO 语句标号

这里允许采用的条件符有

= 等于
<>不等于
< 小于
<=小于或等于
> 大于
>=大于或等于

例如，当希望程序暂停在A的输入为零值的时候，可以使用下列语句来实现

```
20 INPUT A
30 IF A<>0 THEN 50
40 STOP
50 …
```

请注意，在这里

语句标号 STOP

即表示程序在该处暂停运行。

1.2.6 循环

一个程序可用几种方法重复其中某些步骤，其中自动重复某些步骤的程序语句称为循环。执行循环的语句中最简单的是条件转向语句（GOTO语句），其形式是：

语句标号 GOTO 语句标号

例如这个语句就能够同上例中的条件语句一起使用，从而使那个程序继续询问A的值，直到使用者输入零为止。

执行循环的最常见的方法是先用一个设置循环的FOR语句，它的形式是

语句标号 FOR 变量=表达式1 TO 表达式2〔STEP表达式3〕

式中STEP为步长，如遇省略，则表示程序中的步长为1，循环出口则再用一个NEXT语句，其形式为

语句标号 NEXT 变量

FOR和NEXT语句中的变量应是同一个变量，而且变量的值在中间的一些行内不得改变。

例如，当要通过READ语句执行*N*组数据的输入并把这些数据和它们的倒数都显示打印出来，也可以利用循环语句。如

```
10 READ N
20 PRINT "NUMBER","RECIPROCAL"
30 POR I=1 TO N
40 READ A
50 PRNT A,1/A
60 NEXT I
```

循环还能用来生成数据。请考察，例如一个简单的温度换算程序

```
10 PRINT "CENTIGRADE","FAHRENHEIT"
20 FOR C=0 TO 100 STEP 5
30 PRINT C,9*C/5+32
40 NEXT C
```

1.2.7 下标变量

有时，使一个变量在一次程序运行过程中可以取不同的值是非常有助益的（参见例4.3和4.4）。例如，一个程序包括几种材料的数据，那么材料的密度不用变量R1，R2，R3等而称作R（1），R（2），R（3）等显然更方便些，因为这样我们便有可能用一个语句执行关于所有材料的计算，例如程序

```
50 FOR I=1 TO N
60 M(I)=V*R(I)
70 NEXT I
```

便可以根据物体的体积（V）确定每一种材料的质量M（I）。

非下标变量只有一个与其相应的值，而如果使用下标变量就必须有说明所有值的存储单元，这项工作用数组说明语句（DIM语句）加以执行，其形式为：

语句标号 DIM 变量1（整数1）〔,变量2（整数2），…〕

例如

```
20DIMR(50),M(50)
```

这个语句允许多至50个R和M的值，数组说明语句必须出现在首次使用下标变量前。

有些计算机允许使用其它形式的数组说明语句，例如

```
20 DIM R(N),M(N)
```

这里N的值事先已定义。这种形式，当可用时，具有少占存储单元的优点。但本书未采用这种形式。

1.2.8 子程序

有时在一个程序里需要多次使用某一语句系列（例如，见例6.3）。比只重复这些语句更好的方法是把它们放到一个子程序里。这样程序包含的转子语句（GOSUB语

句）如下形式：

语句标号 GOSUB 语句标号

当程序执行到这种语句时，它就岔到（即转移控制）句中第二个语句标号的语句上去。这种语句序列，从该第二个标号的语句开始，而以如下的返回语句（RETURN语句）告终。

语句标号 RETURN

然后程序又返回到紧接着转子语句后面的语句上去。

子程序可放在程序中的任何位置，但把它们放在最后同主程序分开通常更方便。

使用子程序的另一个原因是常常在不止一个程序里包含有这种同样过程（即子程序）。在这样的子程序中最好使用不太常用的变量名（例如用X9而不是用X）。这样能最大限度地避免同一个变量名在一个程序的各个部分代表不同的意义。

子程序也可用来把一个程序分割成若干逻辑单元，象在例7.4中的那样。

1.2.9 其它语句

（1）一些说明性的注释或不需输出的标题都可用以下形式的注释语句（REM语句）把它们放到程序里：

语句标号 REM 注释内容

任何以REM开头的语句对计算机的运行都是不起作用的。有些计算机允许把注记包括在与其它语句同一行内，但本书没有采用这类作法。

（2）非数字的数据（例如字）可用字符串变量加以处理。字符串是一系列在引号内的字母，如“STRESS”，而字符串变量是一个字母后跟随一个$号，例如S$。它们在需要使所显示的标题变化时是特别有用的。

（3）多重转移可用以下控制转向语句加以执行，其形式为：

语句标号 ON表达式 THEN 语句标号1〔，语句标号2，…〕

（在这种语句中GOTO可以代替THEN，如例7.1）和

语句标号 ON表达式 GOSUB 语句标号1〔，语句标号2，…〕

当程序执行到上述一个语句时，如果表达式的整数值是1，它就转移到标号1上去，如果表达式的整数值是2，它就转移到标2号上去，等等。如果表达式给出的值小于1或大于给出的语句标号的数目时就显示出错误信息。

（4）除了已设置在BASIC语言内部的那些标准函数如SIN（X）外，还可以使用自定义函数语句（DEF语句）规定一些自定义函数。例如

10 DEF FNA（X）＝X↑3＋X↑2＋X＋1

定义一个三次函数，这个函数在后来的程序中可作为FNA（变量）加以调用，这个变量的值代替了X，当一个程序中要多次计算某一个代数式的值的时候，可用这种自定义函数（如例7.5）。

1.3 检查程序

大多数计算机当BASIC程序中有语法错误的时候就给出一个清晰的标志，此时一些

程序语句可重新改正，也可以使用专门的汇编程序加以修改。大多数语法错误是容易被查出的，但如果一个变量在程序的不同部分具有不同意义时，可能出现一些莫名其妙的错误。

一个程序仅仅语法正确是不够的。它还必须能给出正确的答案。因此一个程序还应该利用已经有解答的数据加以检查，或者用手算加以检查。如果某程序要在很宽的数据上使用，或者主要不是程序编制员自己使用而是供其他用户使用，那就必须检查程序的所有部分的功能是否准确无误，另一个重要之点是当输入“荒谬”的数据时，要确保程序既不得出错误的也不得出似是而非的答案。要使程序完全避免滥用是相当不容易的，因为这样作程序往往会变得十分冗长。为清楚起见，本书中的程序尽可能保持简短，因此不太可能保证程序对所有输入的数据都得出合理的结果。

1.4 不同的计算机和BASIC 语言的各种变体

本书中诸例所使用的是一种基本的BASIC语言，这种语言对大多数计算机即使储存能力很小的计算机来说都是适用的。程序中每一行只一个单行语句。但是有些计算机允许用分隔符（例如/或：）隔开的方式在一行上书写出几个语句。

各种计算机特别是微型机，一个重要的特征是它的直观显示装置（VDU）如何。它涉及到在屏幕上能同时显示出多大的行幅和列幅。因为某些程序的简单修改可能需要让输出暂时停留在微机屏幕上。

自基本BASIC语言问世以来，相继又出现了各种扩展BASIC语言，并在某些计算机系统中得到实现。如果考虑这些新的成果，本书中的一些程序还可以改写。例如，使用长变量名（如用STRESS代替S或S1）便比较容易书写出清楚的程序。还有其它一些好的方法，例如更有效的循环语句和条件语句以及独立的子程序，这些都使书写结构化程序更容易一些。简言之，结构化程序设计就是使程序格子化，尽量减少语句经由“GOTO 语句标号”和“THEN 语句标号”的转移。良好的程序结构对一些大程序来说是非常有利的。

1.5 BASIC语言简表

赋值语句

LET	计算和赋值
DIM	为下标变量分配存储单元

输入语句

INPUT	“运行时”输入数据
READ	从DATA语句中读数
DATA	数据存储区
RESTORE	使DATA恢复到其始点

输出语句

PRINT	显示输出

程序控制

STOP	暂停程序运行
GOTO	无条件转移
IF…THEN	条件转移
FOR…TO…STEP	开启循环
NEXT	闭合循环
GOSUB	把控制转移到子程序上
RETURN	使控制从子程序返回
ON…THEN	多重转移
ON…GOSUB	多重子程序转移

注释语句

REM	在程序中注释

函数

SQR	平方根
SIN	正弦（用弧度角）
COS	余弦（用弧度角）
ATN	反正切（得出弧出角）
LOG	自然对数（以e为底）
EXP	指数
ABS	绝对值
INT	整数值
DEF FN	自定义函数

1.6 参考文献

1. Alcock, D., *Illustrating BASIC*, Cambridge University Press, (1977).
2. Kemeny, J.G. and Kurtz, T.E., *BASIC Programming*, John Wiley, (1968)
3. Monro, D.M., *Interactive Computing with BASIC*, Edward Arnold, (1974).
4. Sharp, W.F. and Jacob, N.L., *BASIC, An Introduction to Computer Programming using the BASIC Language*, Free Press, New York, (1979).