

GnIMath 2.0 User Guide

2.0.0(*Final*)用户使用手册

功能列表

模块名称	内部功能
表达式计算	1. 计算无变量表达式 2. 求解 $F(x)$ 数值积分 3. 特定数值运算
$F(x)$ 图像绘制	1. 绘制 $F(x)$ 函数图像 2. 求解方程 $F1(x)=0$
$F(t)$ 二维参数曲线绘制	绘制参数曲线 $Fx(t)$ 、 $Fy(t)$ 的图像
曲线拟合及多项式插值	若干样本点的曲线拟合或多项式插值的结果及曲线绘制
色彩深度三维图像绘制	根据颜色深度的三维图像绘制
$F(x, y)$ 三维图像绘制	绘制 $F(x,y)$ 的三维立体图像
$F(t)$ 三维参数曲线绘制	绘制参数曲线 $Fx(t)$ 、 $Fy(t)$ 、 $Fz(t)$ 的三维立体图像
解线性方程组	根据输入的系数矩阵和常数矩阵求解线性方程组或求解矛盾方程组
矩阵计算	矩阵加、减、乘、乘幂、转置、求逆、对应行列式的值的计算求解
线性规划	根据给定的限定条件函数，计算目标函数的最大值和最小值，以及对应的参数

Chapter 1

表达式计算

1. 表达式求值

表达式应当遵循中缀表达式的格式，也就是我们平时所用的表达式记法，“**操作数&操作符&操作数**”的顺序来表示。对于函数应该使用“**函数名 (参数)**”的形式来表示。

对于程序所支持的函数、常量和操作符请参见附录 A。

Example 1.1

计算表达式的值：

$$12 \cos \pi + 2$$

在“**表达式:**”框内输入要计算的不含未知数的表达式

$$12 * \cos(\pi) + 2$$

选择命令“**确定**”，在“**结果:**”框内出现结算结果

$$-10.0$$

表达式:
12*cos(pi)+2

积分区间及步长:
(Xrange)|Step

结果:
-10.0

后退 确定

2. $f(x)$ 定积分数值求解

对于定积分求解这里应当先明白程序的计算原理，以免造成不精准的计算结果的问题。

在程序中使用的定积分数值求解使用的是“复合梯形法”求解定积分。其中 h 为计算步长。

$$\int_a^b f(x) dx \approx \frac{h}{2} [f(a) + 2 \sum_{i=1}^{n-1} f(ai + h) + f(b)]$$

数值解法一般都是由误差的，复合梯形法的误差为

$$-\frac{1}{12}(b-a)^2 h^2 f''(\xi)$$

虽然理论上 h 越小越好，但是受到计算精度的影响则并非如此。

原因是在计算机中大数和小数加减运算会使小数丧失精度。

Example 1.2

计算定积分的值：

表达式:
x*x

积分区间及步长:
(Xrange)|Step
(0,5)|0.0001

结果:
积分为41.666666675050465

后退 确定

$$\int_0^5 x^2 dx$$

在“表达式:”框内输入积分表达式

$x*x$

在“积分区间及步长:”框内输入积分区间和计算步长

$(0,5) | 0.0001$

选择命令“确定”，在“结果:”框内出现结算结果

积分为 41.666666675050465

在例 1.2 中的计算结果与准确解 41.6666666.....误差小于 1E-7，但是当选取计算步长为 0.00001 时计算结果为 41.666916666880596!这个结果的误差大于 1E-4!甚至还不如选取步长为 0.001 的情况，该结果为 41.66666749999568。所以建议不要使用该功能计算较高精度的计算，结果受截断误差的影响很大!!!

3. 特定数值运算

这里提供了几种常用的运算，见下表

功能	指令	示例	示例说明
组合数	c	c 5,3	计算组合数 c(5,3)----10.0
排列数	a	a 5,3	计算排列数 a(5,3)----60.0
阶乘	!	! 10	计算 10 的阶乘----3628800.0
判定素数	iss	iss 13	判定 13 是否为素数----true
求余数	%	% 12,5	计算 12 除以 5 的余数----2
因数分解	d	d 120	因数分解 120---2*2*2*3*5
弧度转换为角度	r2d	r2d 3.14	把弧度 3.14 转化为角度----179.908...(省略)
角度转换为弧度	d2r	d2r 180	把角度 180 转化为弧度----3.1415...(省略)

示例 d|120 的截图见右图。



Chapter 2

F(x) 图像绘制

1. 函数 $f(x)$ 图像绘制

这个功能与常数表达式计算都是以前写 GniMath1.0 的时候的老功能了，其中计算表达式的部分全部重写了一遍，因为以前的计算写的是堆栈分析表达式并一次求解的程序，所以在每次计算的时候都要再次扫描表达式字符串，效率很低，表达式一长的时候扫描的时间延迟就体现出来了。这次使用的是二叉树结构保存了扫描完成的表达式，也就是仅仅在读入表达式的时候扫描一次，然后计算的时候就只需要遍历树就可以了。其他部分的代码基本都是沿用以前的，所以使用上和老版本差不多，但是有细微的易用性的修改。（对我自己而言，上次写纯粹把 Java 当 C 用了，这回终于面向对象了，专门写了个 World2D 类）

这次的输入数据可以为任意多个表达式（因为现在计算效率提上去了），但是建议不要画 5 条以上，感觉很乱，毕竟手机屏幕小.....其他的输入也改到了一个文本框内，看着舒服点，当然我就得自己写代码检查和扫描了！

另外，这次加入了颜色信息，可以自己设定绘制曲线的颜色，颜色信息写到方括号内，三个分量分别为红(Red)、绿(Green)、蓝(Blue)，值为 0 到 255 之间的整数。如果不输入颜色信息则默认为红色。关于颜色更多信息参考附录 B。

Example 2.1

绘制函数图像：

$$f_1(x) = \sin(x)$$

$$f_2(x) = \cos(x)$$

在“表达式：”框内输入要绘制的函数表达式

（别忘了结束符----分号！）

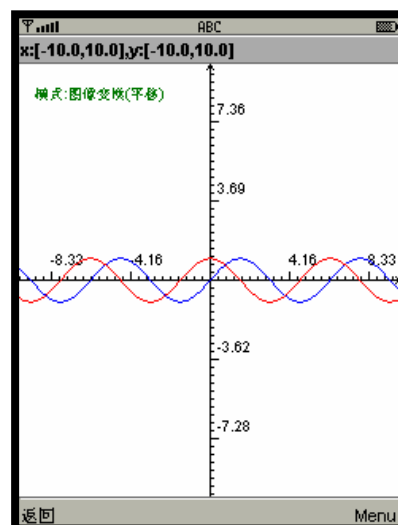
[0,0,255]sin(x);cos(x);

在“绘制范围：”框内输入绘制的 x、y 的显示范围

(-10,10) | (-10,10)

选择命令“确定”即得到绘制结果界面

※对于坐标变换的功能键设定：



	模式：图像变换（平移）	模式：图像变换（缩放）
选择键(Select)	切换到缩放模式	切换到平移模式
上(Up)	上移 1/4 屏幕	水平方向缩小 1 倍
下(Down)	下移 1/4 屏幕	水平方向放大 1 倍
左(Left)	左移 1/4 屏幕	竖直方向放大 1 倍
右(Right)	右移 1/4 屏幕	竖直方向缩小 1 倍

※对于有触摸屏的手机，就可以享受到一个很爽的功能了，也就是显示坐标。嗯，程序是自用.....其实功能基本都是给我的 Moto-Ming 定制的，所以大家会发现我根本没有用任何按键输入.....- -!

Example 2.2

绘制函数表达式：

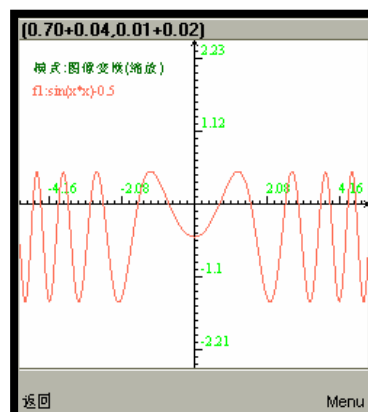
$$[255,127,127]\sin(x*x)-0.5;$$

调整到一个合适的位置，这里点击了正半轴最近的零点，
Look 以下上面标题！

$$(0.70+0.04,0.01+0.02)$$

这就是选中坐标的点，加号后面是可能误差。

注意：加号代表正负误差，也就是说 $X=0.70\pm 0.04$ 的范围内！



2. 求解方程

还是先说明算法的计算理论，让大家用的时候心里明白怎么回事比较好。根据零点定理，如果函数 $f(x)$ 在闭区间 $[a,b]$ 上连续，且函数值 $f(a)$ 、 $f(b)$ 异号，则在 (a,b) 内至少存在一点 ξ ，使得 $f(\xi)=0$ ($a<\xi<b$)。这点也就是 $f(x)=0$ 的根！

当最早的 GniMath1.0 做完以后，我自己拿 Plot2D 一般不是看函数，而是解方程，这不是最早预想的功能，因为坐标的标题都只有 2 位小数的精度，我把图像放大到一定水平，只能得到 2 位小数精度的根，于是这次，重写的时候，我加入的解方程的功能。

Example 2.3

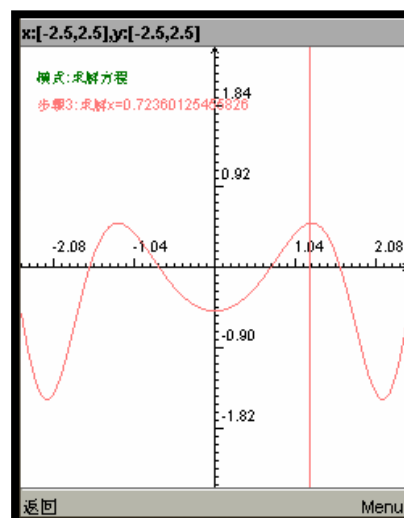
使用例 2.2 中的图像来做演示，先选“求解方程 f1”
然后选择求解下限，这里可以使用左右按键来移动
然后点击 Select 确定，本例选择的下限为原点左右，
上限为正半轴第一个峰值左右。然后 Select 确定得到最终解。

$$0.72360125455826$$

在计算中应有 $\text{Right-Left} < 1\text{E-15}$ ，所以有 15 位的精度

这里输出了 14 位，因该够用了吧！? ☺

用例 2.3 的计算结果和 2.2 的显示比较一下，嗯.....



注：实际使用中还会有这样的情况，比如多个零点的区间，实际只能解出一个根，所以自己选区间的时候注意把。还有，如果选了象 \tan 那种无穷两边的正负区间，可以解出来那个无穷点。

Chapter 3

二维参数曲线绘制

曲线绘制

该参数曲线的绘制方式与函数 $y=F(x)$ 的绘制方式类似，但是这里 t 的步长 dt 很重要，如果 t 的步长选低了会造成锯齿，选高了的话.....刷新慢就是了.....

Example 3.1

要绘制的参数曲线：

$$\begin{cases} \frac{30t}{1+t^3} \\ \frac{30t^2}{1+t^3} \end{cases} \quad \begin{cases} \frac{15t}{1+t^3} \\ \frac{15t^2}{1+t^3} \end{cases}$$

在“表达式:”框内输入要绘制的函数表达式

$[0,0,255](30*t)/(1+t*t*t) | (30*t*t)/(1+t*t*t);$

$(15*t)/(1+t*t*t) | (15*t*t)/(1+t*t*t);$

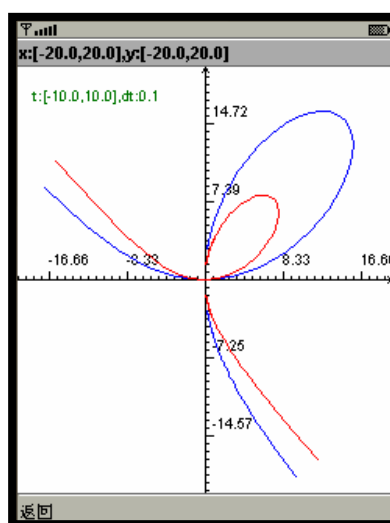
在“参量 t 范围:”框内输入绘制的 t 的范围和计算步长

$(-10,10) | 0.1$

在“绘制范围:”框内输入绘制的 x 、 y 的显示范围

$(-20,20) | (-20,20)$

选择命令“确定”即得到传说中的笛卡儿叶形线
点手机选择键 (Select)，可以切换显示出输入的表达式。



Chapter 4

曲线拟合及多项式插值

1. 关于曲线拟合和多项式插值

根据多项式插值定理：若 x_0, x_1, \dots, x_n 是不同实数，对于任意数值 y_0, y_1, \dots, y_n ，存在唯一的次数之多是 n 次的多项式 p 使得 $p(x_i)=y_i$ 。这里我们就是寻找这个通过这些样本点的 n 次多项式。

本程序的算法使用的是均差算法，具体的内容可以去参考任何一本计算方法或者数值分析教材，一般都是有的。

曲线拟合的方式根据解矛盾方程组得最小二乘解来获取一条最佳的曲线，也就是离各个样本点最近的曲线。（0 次的时候相当于求平均值，1 次的时候相当于线性回归），在计算 $N-1$ 次的曲线时其实就相当于计算插值多项式，不过这里如果次数是 $N-1$ 则按均差的方法来求解插值多项式。

2. 曲线绘制及计算结果

首先我们需要输入一组样本点，满足格式 “ $...(x_n, y_n);...$ ”，并且不应有重复的 x 值出现。

Example 4.1

在“样本点数据:”框内给出样本点

(3,1);(1,-3);(5,2);(6,4);

在“拟合次数:”框内给出次数

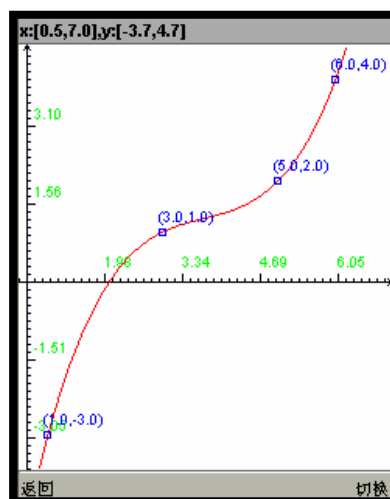
3

选择命令“确定”即得到插值曲线

框线蓝色圈住的为样本点位置

点手机选择键(Select)可以显示出样本点坐标

选择“切换”命令可以得到均差值和多项式系数



对于次数比样本点减一小的情况，将进行曲线拟合

拟合的曲线不会通过各个样本点（不一定……），仅仅是离各个样本点最近的指定次数曲线。

我们用上面的样本点再来做一个曲线拟合的示例。

Example 4.2

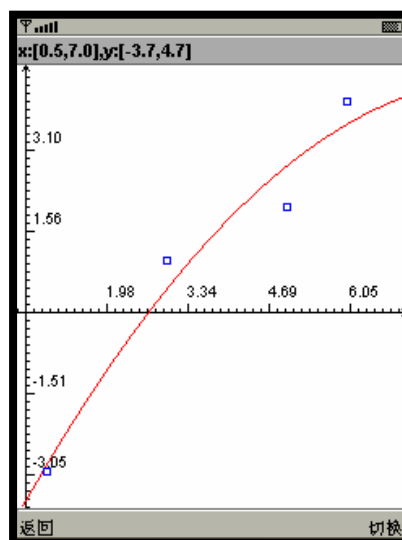
在“样本点数据:”框内给出样本点

(3,1);(1,-3);(5,2);(6,4);

在“拟合次数:”框内给出次数

2

选择命令“确定”即得到拟合的曲线



Chapter 5

深度色彩三维图像绘制

1. 程序构想

在原来看图形学书的时候写过一个 GLUT 的色彩深度程序，最后觉得效果还行，至少能在平面上表示出第三维，只不过把 Z 的值转化成了颜色，对于人眼其实也是很容易能分辨的。所以这次做功能扩展的时候就加了进来。

这里使用两个颜色分量来分辨---红色和蓝色。计算的时候其实只有一个 Z 值比例，具体的分配方式使用 $[z, 0, 255-z]$ ，所以对于越大的 Z 值，颜色越红，反之越蓝。这里通过输入 Z 的范围来计算这个参数 z。（毕竟手机计算速度确实不行，如果先扫描计算一次 Z 的范围要慢 2 倍的）

Example 5.1

绘制表达式图像：

$$\sin(x) \cos(y)$$

在“表达式 f(x,y):”框内输入要绘制的函数表达式

$$\sin(x) * \cos(y)$$

在“绘制范围:”框内输入绘制的 x、y 的显示范围

$$(-4,4) | (-4,4)$$

在“参数:”框内输入 z 的绘制范围和绘制像素步长

$$(-1,1) | 5$$

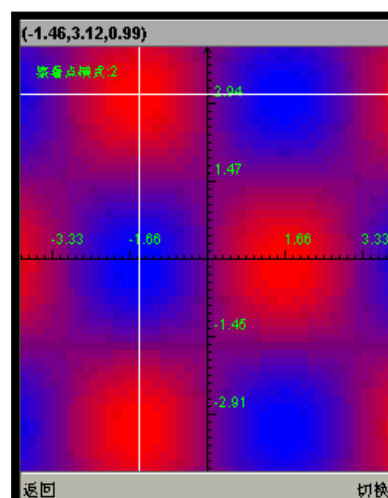
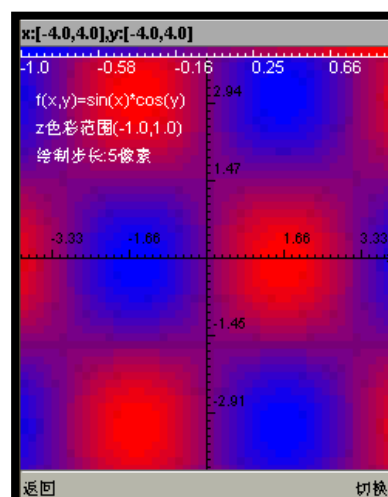
选择命令“确定”即得到图像

点手机选择键(Select)可以显示绘制信息，上面的色彩条是 z 数值变化的颜色参照轴

对于有触摸功能的手机可以直接点选某一点来显示该点的(X,Y,Z)坐标，如果没有，作为弥补，我写了那个切换的功能，等价于前者，不过是通过上下左右来移动选择要查看的点。察看点模式分两种，一种一次走 20 像素，另外一种 2 像素。同过点选 Select 来切换。

当计算会超出输入 z 范围的时候将会在超出的范围点显示为其他的颜色。

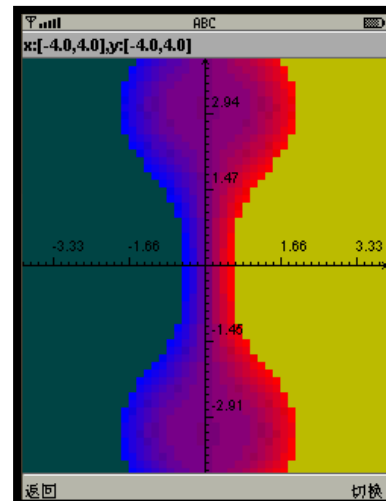
Example 5.2



示例函数表达式：

$$\sin(x) * \cos(y) + x$$

其他参量与上面相同，选择命令“确定”
深色的部分是超出下界的，浅色的部分是超出上界的。



Chapter 6

三维图像绘制

1. 绘图原理

对于给定的 x 、 y 范围，可以计算出该范围内的很多个 z 点，程序选取均分的方式，截取区域，然后计算出对应的 z 值数组。（同时也得到了 z 的最大值和最小值）然后按照相邻的 4 点连线的方式绘制出三维网格。（需要 3 维到 2 维的坐标转换公式，大家可以参考解析几何的书）

因为手机计算速度和资源实在和计算机没法比，我也没想到时间和空间复杂度比较低的染色算法（参考了几个计算机图形学的书，那些算法要么耗时，要么耗空间），有哪位高人能提供什么牛叉染色算法的话，请联系我。（不过说实在的，现在这样仅绘制网格，如果精度高了都会有延迟，再改确实比较困难，毕竟计算也是耗时的）

2. 图像绘制

Example 6.1

绘制函数：

$$\sin(x) \cos(y)$$

在“表达式 $f(x,y)$ ：”框内输入要绘制的函数表达式

$$\sin(x)*\cos(y)$$

在“绘制范围：”框内输入绘制的 x 、 y 的计算范围

$$(-4,4) | (-4,4)$$

在“平移及缩放比：”框内输入绘制像素的 x 、 y 偏移和图像放大比例

$$(-0,0) | 20$$

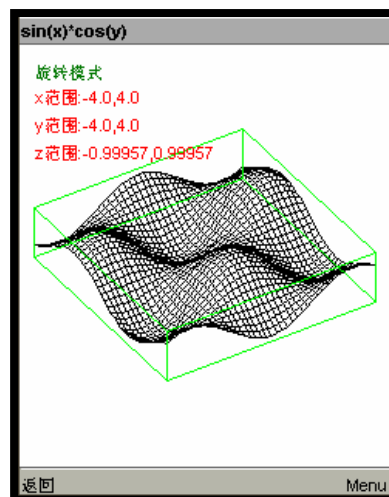
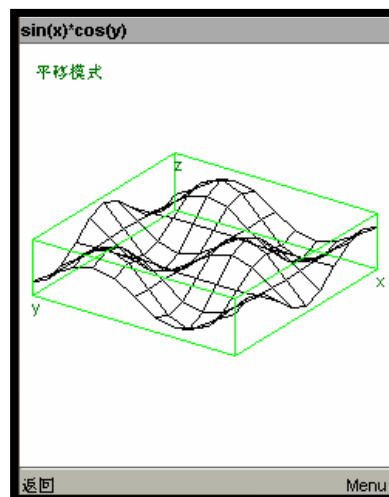
选择命令“确定”即得到图像

点手机选择键(Select)可以显示范围信息和切换模式
平移模式用于平移图像，旋转模式用于调整视角，操作都是通过上下左右完成

如果是触屏，嘿嘿，有个超好玩的拖动旋转功能！

可以用的话试试就知道了！~

右图是提高精度的图像，转了转~呵呵



Chapter 7

三维参数曲线绘制

1. 绘图原理

程序所使用的坐标影射和上面的三维图象绘制是一样的。(自己好不容易一次面向对象, 顺便给弄了个 World3D 类) 不同的是, 这个绘制的时候是按照参数 t 从起始, 每次走特定步长, 直到 t 的终止点。每次连接相邻的两个点得到的图像。

2. 图像绘制

Example 7.1

要绘制的曲线方程:

$$\begin{cases} \sin(t) \\ \cos(t) \\ 0.1t \end{cases}$$

在“表达式:”框内输入要绘制的函数表达式

$\sin(t) | \cos(t) | t*0.1$

在“参量 t 范围:”框内输入参量 t 的范围和计算步长

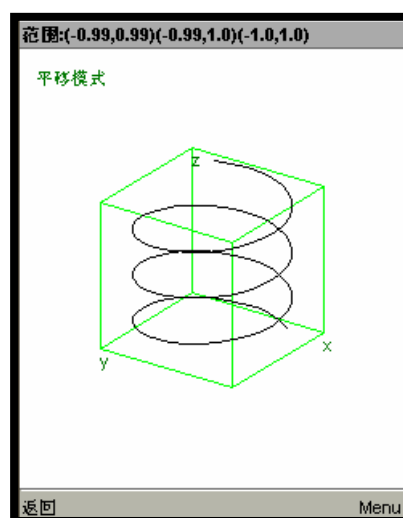
$(-10,10) | 0.2$

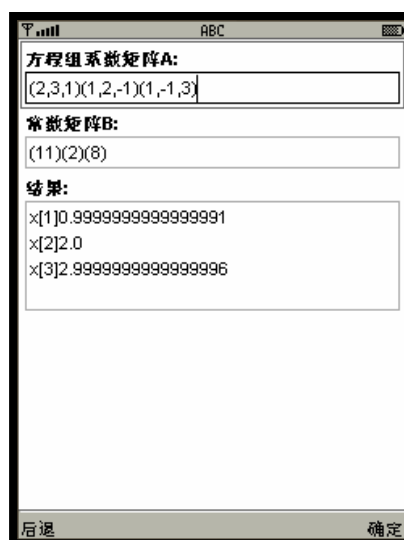
在“平移及缩放比:”框内输入绘制像素的 x 、 y 偏移和图像放大比例

$(-0,0) | 50$

选择命令“确定”即得到图像

操作方式与三维函数图像图像操作完全相同。这里在标题栏显示了 x 、 y 、 z 的范围。因为有些手机的显示范围小, 标题栏可能 z 的范围就显示不出来了, 所以可以在菜单里找到“切换”命令, 进去会看到相关计算信息和参量, 点击选择键(Select)可以切换信息(信息分两个屏, 需要切换)





选择命令“确定”得到计算结果

X[1]0.9999999999999991

X[2]2.0

X[3]2.9999999999999996

标准结果应该是 1、2、3，所以结果精度还是可以的，自己保留小数四舍五入吧。

2. 求解矛盾方程组

Example 8.2

要计算的矛盾方程组：

$$\begin{cases} x_1 = 1 \\ x_2 = 2 \\ x_3 = 3 \\ -x_1 + x_2 = 1 \\ -x_2 + x_3 = 2 \\ -x_1 + x_3 = 1 \end{cases}$$

在“方程组系数矩阵 A:”框内输入系数矩阵

(1,0,0)(0,1,0)(0,0,1)(-1,1,0)(0,-1,1)(-1,0,1)

在“常数矩阵 B:”框内输入常数矩阵

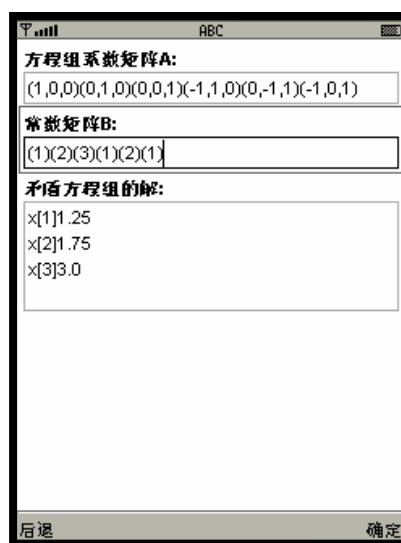
(1)(2)(3)(1)(2)(1)

选择命令“确定”得到计算结果

X[1]1.25

X[2]1.75

X[3]3.0



Chapter 9

矩阵计算

1. 计算原理

实现的算法加减乘都是按定义做的。对应行列式的值采用模拟消去化为上三角行列式然后对角线元求积得到。逆矩阵用的也是模拟消去的办法用另一个单位阵通过矩阵初等变换计算而来，矩阵的乘方使用了一个保存部分计算两的 2 倍乘幂向上的算法，最多计算 62 次矩阵乘法就可以算到最大的整数次幂。

2. 指令表

输入数据采用 “**Command | Data**” 的方式来执行操作

指令 Command	参数 Data	功能
Add	矩阵信息	添加矩阵，返回编号
Del	矩阵编号	删除矩阵
Save	无（有则忽略）	保存结果矩阵，返回编号
Show	矩阵编号（0 代表结果矩阵）	显示矩阵
Com	+(矩阵 1 编号, 矩阵 2 编号)	矩阵加法
...	-(矩阵 1 编号, 矩阵 2 编号)	矩阵减法
...	*(矩阵 1 编号, 矩阵 2 编号)	矩阵乘法
...	^(矩阵 1 编号, 幂次)	矩阵乘幂
...	R	矩阵求逆
...	T	矩阵转置
...	D	方阵对应行列式的值

3. 计算矩阵

这里给出一个计算示例，列出所有指令，然后给出结果图

Example 9.1

在“命令及参数:”框内输入如下指令序列（每行一个）

该示例演示 8.1 的方程求解

add | (2,3,1)(1,2,-1)(1,-1,3)

add | (11)(2)(8)

com | r(1)

save |

com | $\ast(3,2)$

得到结果如图所示



Chapter 10

线性规划

1. 计算原理

由于这部分的代码不是我写的，我也只能大概给介绍一下算法，这里使用的算法是线性规划较为常用的（或者说是古典的）单纯形法来求解，找寻边界点来计算极值。（这种方法将图形转化为凸性的，所以极值点一定在边界点上）详细的什么标准型以及松弛性转换，这里就不做什么介绍了……中学阶段所学习的线性规划仅限于 2 维，而这里的线性规划可以是任意维。单纯形虽然不是一个多项式时间的算法，但是在实际通常相当快。

2. 问题求解

Example 10.1

这里求解问题的约束条件为：

$$\begin{cases} x_2 \geq 2 \\ 2x_1 - x_2 \geq 4 \\ x_1 + x_2 \leq 10 \end{cases}$$

目标函数为：

$$z = -2x_1 + -x_2$$

这里需要先将所有约束条件小于等于化：

$$\begin{cases} -x_2 \leq -2 \\ -2x_1 + x_2 \leq -4 \\ x_1 + x_2 \leq 10 \end{cases}$$

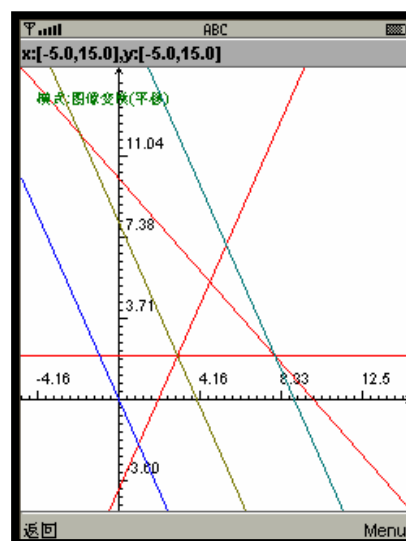
右图通过表达式绘制给出了约束条件的区域范围及目标函数。红色（约束条件），蓝色（目标函数），另外两条是最大和最小值结果。

在“约束条件系数：”框内输入变换的约束条件系数

(0,-1)(-2,1)(1,1)

在“约束条件常数：”框内输入变换的约束条件常数

(-2)(-4)(10)



在“目标函数系数:”框内输入变换的约束条件常数

(-2,-1)

最后选择“确定”得到结算结果:

最大值:-8.0

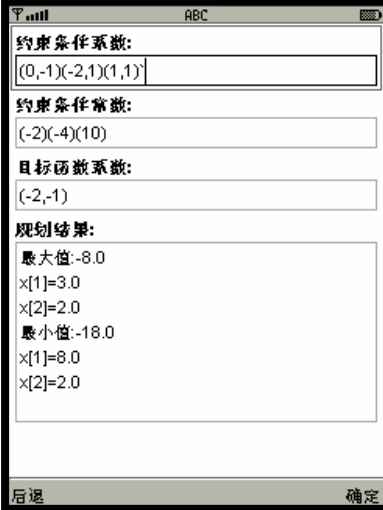
X[1]=3.0

X[2]=2.0

最小值:-18.0

X[1]=8.0

X[2]=2.0



The screenshot shows a window titled "ABC" with a toolbar containing a calculator icon and a help icon. The window contains several input fields and a results section:

- 约束条件系数:** (Constraint coefficients) with the input $(0,-1)(-2,1)(1,1)^T$.
- 约束条件常数:** (Constraint constants) with the input $(-2)(-4)(10)$.
- 目标函数系数:** (Objective function coefficients) with the input $(-2,-1)$.
- 规划结果:** (Optimization results) displaying:
 - 最大值:-8.0 (Maximum value: -8.0)
 - x[1]=3.0
 - x[2]=2.0
 - 最小值:-18.0 (Minimum value: -18.0)
 - x[1]=8.0
 - x[2]=2.0

At the bottom of the window, there are two buttons: "后退" (Back) and "确定" (Confirm).

Appendix A

函数、常量及计算符

1. 支持的常量

Pi 3.141592653589793

E 2.718281828459045

2. 支持的函数

函数名	功能	备注
Sin	正弦函数	Math 库函数
Cos	余弦函数	Math 库函数
Tan	正切函数	Math 库函数
Abs	绝对值函数	Math 库函数
Asin	反正弦函数	自定义函数精度 1E-15
Acos	反余弦函数	自定义函数精度 1E-15
Atan	反正切函数	自定义函数精度 1E-15
Exp	底数为 E 的指数函数	自定义函数精度 1E-20
Log	底数为 E 的对数函数	自定义函数精度 1E-20
Sqrt	开方函数	Math 库函数

注：紫色的函数为 *g2mE* 库中没有的函数，自己实现的，后面写了函数的计算精度

3. 支持的运算符

运算符	功能	备注
+	加法	A+B 等价于 A+B
-	减法	A-B 等价于 A-B
*	乘法	A*B 等价于 A×B
/	除法	A/B 等价于 A÷B
^	乘幂	A^B 等价于 A^B

Appendix B

关于颜色

1. 表示方式

颜色使用 RGB 分量来表示，输入的时候放在方括号内，即[**R**, **G**, **B**]，每个分量越大，对应的颜色量就越多，比如红色为[**255**,0,0]，即全部是红色。

2. 一些颜色示例

[0,0,0]	黑色		[255,255,255]	白色	
[255,0,0]	红色		[127,0,0]	深红	
[0,255,0]	绿色		[0,127,0]	深绿	
[0,0,255]	蓝色		[0,0,127]	深蓝	
[255,255,0]	黄色		[127,127,0]	深黄	
[0,255,255]	浅蓝		[0,127,127]	海绿	
[255,0,255]	粉红		[127,0,127]	紫色	

根据上面给出的颜色，大概就可以推测其他颜色了.....

Appendix C

关于作者

Name: *Ming*

E-mail: fe3000@qq.com

贡献者名单:

ZhangZhen(mfzz1134@163.com)