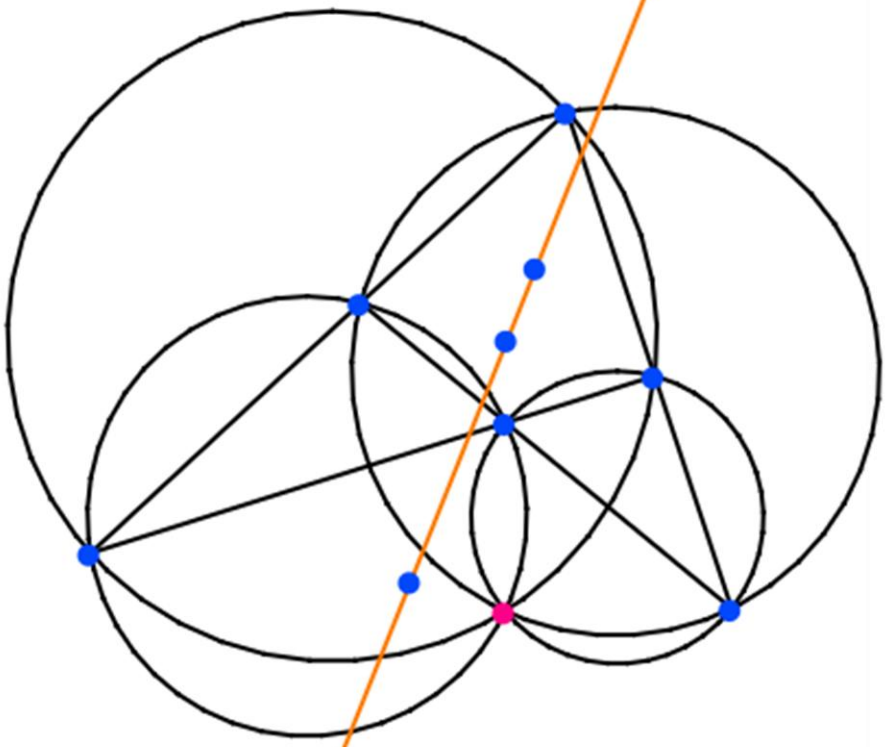
Version 4.12+



稽何计算器

用户说明

第 18 次修订

SQY

一起塑造 SGC 的未来

SGC v4 基于 Scratch 构造，使用 Scratch 制作此类软件实属不易。由于作者能力有限，SGC 一定还存在许多缺陷。若您在使用过程中遇到问题，还请及时和 SQY 反馈。

SGC v5JavaScript 版本正在制作中，敬请期待 SGC v5！

使用须知

感谢您对 SGC 感兴趣。本软件使您能够方便地探索数学。您可以使用 SGC 来绘制函数、创建动画、编辑程序、进行几何构造等。

若要使用 SGC，您同意如下条款：

a. SGC 是开源软件. 您不对 SGC 进行无意义的二次改编或者将 SGC 的代码据为己有.

b. 不对 SGC 官网发送、存储无意义的信息.

c. 不得将 SGC 用于：

i. 违反中华人民共和国宪法、其他法律法规以及其他您所在地方的法律法规;

ii. 以计算器为媒介传播非法内容（包括但不限于暴力、色情、淫秽、辱骂、骚扰、侵权内容）;

iii. 使用此计算器进行作弊等行为（SQY 对因此产生的后果不承担任何责任）;

d. 将 SGC 用于您个人的非商业用途, 在为商业用途前请先申请。

这些条款可能会进行修订、增补。您在条款更新后继续使用 SGC，即表示您同意修订后的条款。如果不同意，您必须停止使用 SGC。SGC 保留在通知或不通知您的情况下修改或终止使用 SGC 的权利。

SQY 版权所有。本软件源代码开放，可以自由修改，使用源代码需要标注原作者。不得售卖源代码。

2026.1

简介

稽何计算器（SQY Graphic Calculator，下称 SGC）是一个数学软件，为通过 Scratch 探索数学世界、领略数学之美而生。SGC 始于 2022，在 Scratch 以及能支持 Scratch3 的平台上运行。

SGC 功能丰富，可以绘制显函数、隐函数、参数方程、极坐标等多种类型的图像，也可以进行简单的数据统计与分析，创建简单的几何对象。在 TurboWarp、HTML 环境下，运行速度与 desmos、GeoGebra 接近。您可以自己设置参数，以获得比 desmos 更好的绘图效果。

SGC 兼容性高，在手机、平板、电脑上都可以运行。

目前最新的版本是 4.10。4.5 以上的版本能在 SGC 内获取更新。

敬请期待 SGC v5!

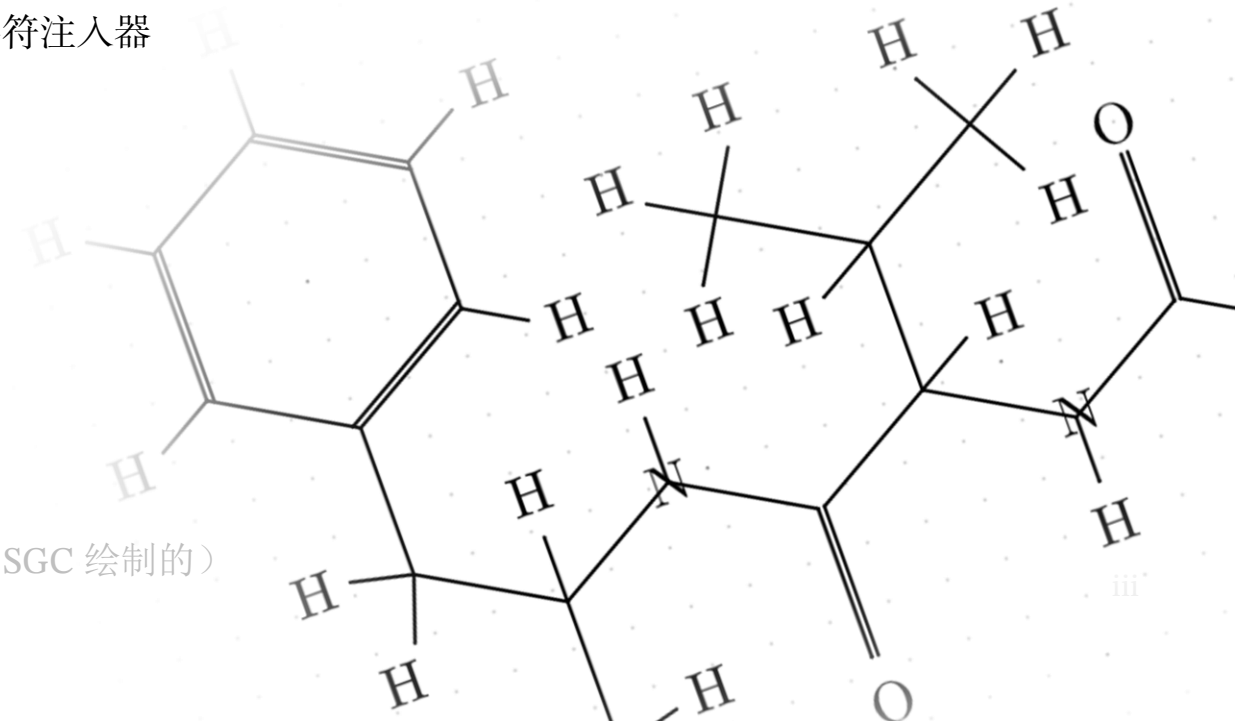
访问 https://sqy419.axolotlpower.com/sgc 以获取更多信息。

致谢：

1. turbowarp.org; githubpages; gtd232(美西螈力量)
2. Obdopqo-四万字超大画笔字库代码以及示例字库
（使用教育部-通用规范汉字表，约减后收录 10000 余字符）
3. Alpha Math、Origin 图形计算器：同类优质作品，交流互鉴
4. “6”的科学计算器中的公式渲染器、幂的计算及傅里叶变换音乐生成器
5. 其他对 SGC 的开发做出贡献的开发者

使用了如下非原创内容：

1. CMU Sans Serif、CMU Serif、JetBrains Mono 等字体
2. Obdopqo 多边形填充与点击判断、四万字超大画笔字库
3. -Rex- SFE 字符注入器

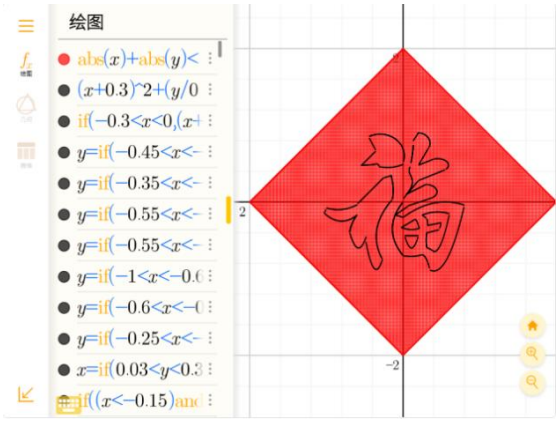


（这也是由 SGC 绘制的）

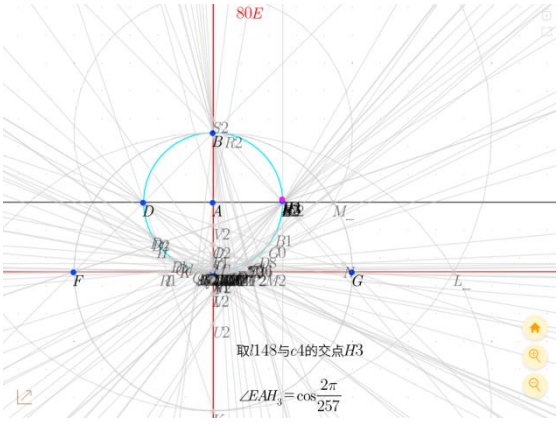
使用 SGC,探究数学之美

Enjoy math with SGC

轻松创建创意图表，分享你的想法！

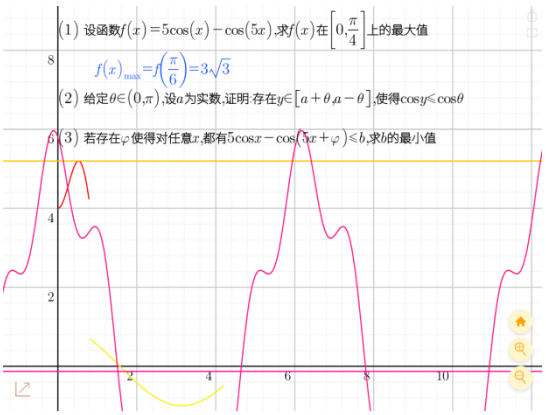


使用脚本，打破常规，创建高级演示动画，易如反掌。



不善言辞，如何表达情感？面对喜欢的人，总不能只用乏味的词语。SGC 帮助你使用数学之美，win his/her heart.

使用 SimpleSTeX，显示优雅公式与图像的结合。



常见问题

1. 为什么画不了图像？

   a. 可能是你没有点 enter。只有将函数输入后，点击 enter 或者虚拟键盘上的箭头，才能解析输入的内容并绘制。目前 SGC 没有预览功能。

   b. 可能是你输入的函数计算比较复杂，需要解析较长时间，或者设置精度较高的隐函数。

   c. 出现 bug。

2. 为什么我保存的东西没法导入 Arch/Alpha Math？为什么 Arch/Alpha Math 计算器里面的某些东西没法导入进来？

   有些函数语法不一样。目前 SGC 只能导入自己的存档。

3. 为什么画出来的隐函数效果很差，丢失细节？

   （参见章节 10.2）

   这有可能是精度不足。在设置选择“绘图”，在当前模式中点击“隐式精度”，输入合适的精度。精度过高可能会导致软件崩溃。

   精度很高时，建议不要在绘制中进行其它操作，否则可能会导致计算错误。

   拖动网格时可能会终止正在进行的计算。

4. 如何限制函数的定义域？

   使用 if 函数。例如想要限制 $y=-x+23$ 在 $x \in (0,15)$ 上，可以使用 if(0<x<15, -x+23)。注意：如果使用 $y=-x+23(0<x<15)$ 则不能得到希望的图形，因为这表达式等价于 $y=-x+(23(0<x<15))$，限制定义域的那一部分只作用在了 23 上。

5. 为什么 $y=\frac{x^{x+1}}{(x+1)^x}$ 在 x 比较大的时候无法绘制？

   虽然 $\lim_{x\to\infty}\frac{\frac{x^{x+1}}{(x+1)^x}}{x}=\frac{1}{\mathrm{e}}$，但是 x^{x+1}、$(x+1)^x$ 在计算过程中溢出了，导致计算变为 ∞/∞，因此无法绘制。这一类问题还表现在 $y=3^x\sin x$、$y=\ln x^x$ 上。

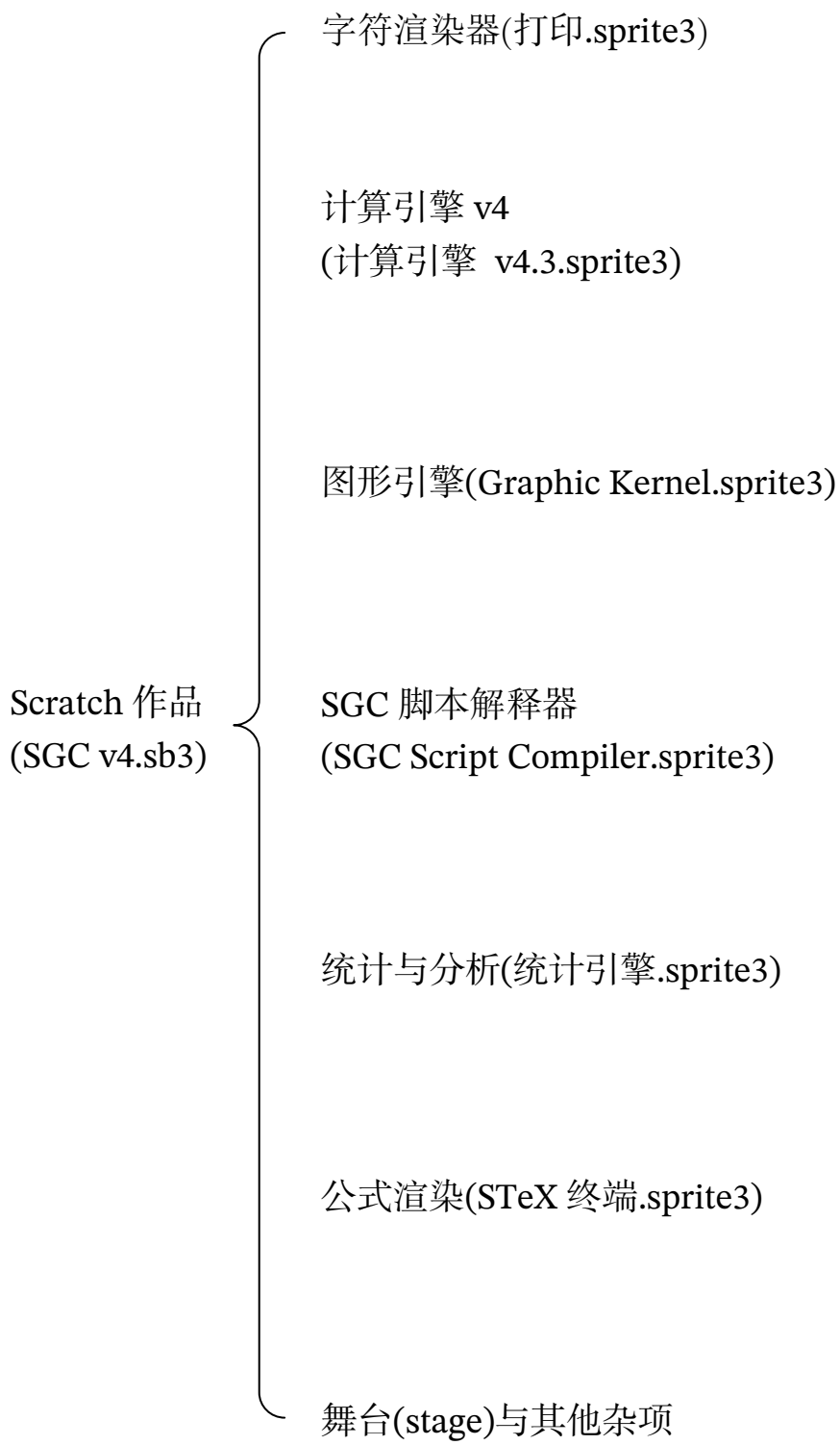
6. 为什么 $y=\ln(\sin x+\cos y)$ 在 $y<0$ 的渐近线绘制不出来？

   这是由于 ln$x$ 在 0 附近，值由 NaN 变为 $\lim_{y\to-\infty} y$ 导致的。关闭“跳过未定义值”选项即可。

快速引导

1. 想要渲染 LaTeX 公式？ （参见 7.2 STeX）
2. 想要保存/加载文件？ （参见 1.2 基本操作）

本软件架构



您可以添加其他角色以满足自己的需求。

目录

1 主界面

1.1 界面介绍
1.2 菜单栏
1.3 键盘
1.4 素材库

2 输入

2.1 输入方式
2.2 输入类型
2.3 输入语法
2.4 创建对象

3 输入对象操作

3.1 主界面操作
3.2 小窗口操作

4 坐标系

4.1 平面坐标系
4.2 3D 坐标系

5 计算引擎

5.1 技术参数
5.2 计算
5.3 运算符与函数
5.4 高级运算符
5.5 构造指引

6 几何

6.1 基本操作
6.2 创建几何对象
6.3 轨迹

6.4 修改对象样式

7 文本

7.1 普通文本
7.2 高级文本
7.3 STeX

8 脚本

8.1 脚本编辑器
8.2 语法
8.3 操作 SGC 底层
8.4 系统常数
8.5 图形库
8.6 动画播放

9 统计

9.1 统计表概述
9.2 数据的输入
9.3 数据的显示
9.4 列操作
9.5 列统计计算
9.6 列回归计算

10 设置

10.1 全局
10.2 坐标系与网格
10.3 绘图
10.4 性能
10.5 实验功能与其他
10.6 变量设置

11 多边形、逐帧动画与批注

11.1 多边形
11.2 动画

11.3 批注

12 附录

12.1 示例表达式
12.2 几个软件对比

13 关于内部实现的一些说明

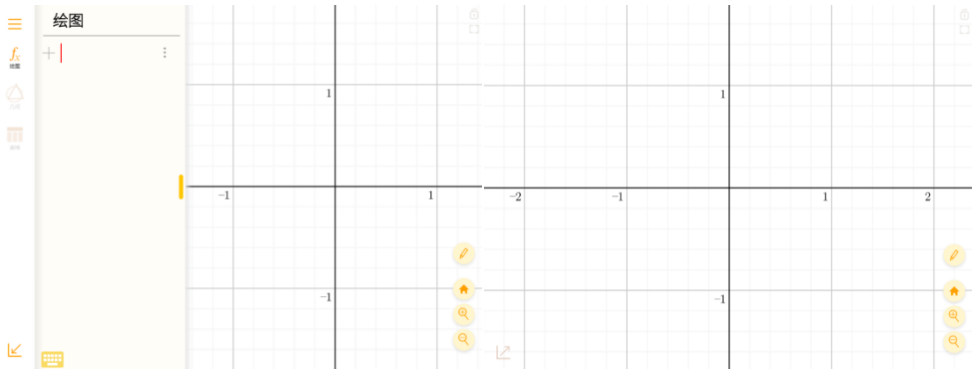
13.1 Scratch 的数据类型
13.2 提升三角函数的计算精度
13.3 数值一阶导数
13.4 作函数的图像
13.5 数值断点判断
13.6 数值积分
13.7 一元方程的数值解与 fSolve 函数
13.8 特殊函数的数值计算

约定

1. Infinity 必须大写。小写的 infinity 不是 Infinity。
2. 在计算中，true=1，false=0。
3. 计算中，常数 Pi 和 EulerGamma 也可以不大写。但是 Gamma 不指 EulerGamma，而是指 Gamma 函数。
4. SGC 标准颜色代码：aabbccdd。aa 填入颜色，bb 填入饱和度，cc 填入亮度，dd 填入透明度。aa、bb、cc、dd 范围均为 0 至 99。如 00999900 是相当于#ff0000 的红色。
5. v4.8 以下的 SGC 的像素坐标系范围: x=-240 ..240 ,y=-180 ..180。对于 v4.8 以上的 SGC，具有自适应调整分辨率的功能，可以使用设置-显示缓存查看分辨率。
6. NaN 会显示为"?"。
7. ≤、≥需使用<=、>=替换。

1 主界面

1.1 界面介绍

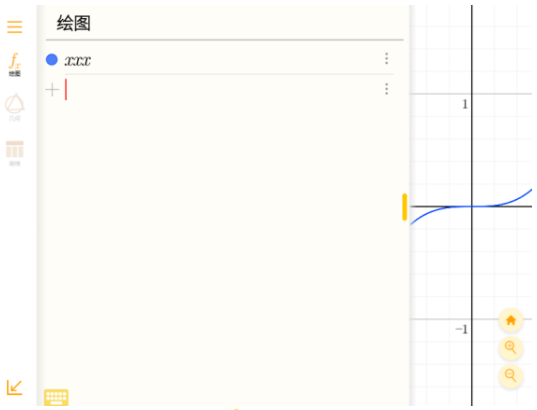


Scratch 用户可以点击小绿旗以启动计算器。启动计算器之后，会有如图的界面。这就是主界面。

左侧浅黄色区域为输入区，右边有网格的区域是图形区。最左边白色区域是模式区。点击汉堡按钮“ ≡ ”可以打开菜单。点击“ f_x 绘图 ”“ 几何 ”“ 表格 ”切换到对应的模式。

左下角的按钮“ ↙ ”可以隐藏输入区，仅显示图形，点击后这按钮变成了“ ↖ ”；再次按下此按钮可以显示输入区。

按下右侧小房子图标可以使坐标系回到初始位置，使用右侧放大镜图标可以放大或者缩小坐标系。

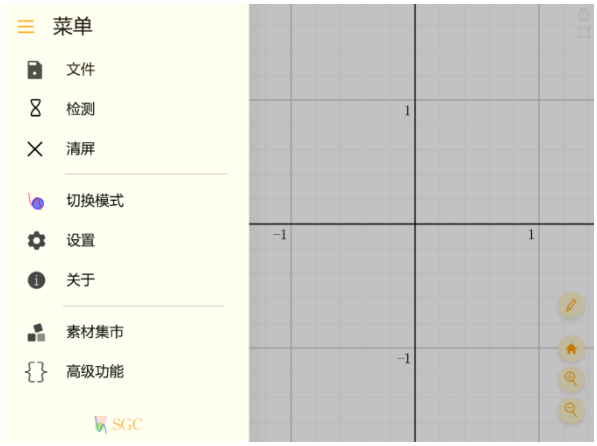


在函数显示区,你可以拖动右侧黄色“ ”来更改显示区的宽度。左下方有一个黄色的键盘图标“ ”，点击即可打开虚拟输入键盘。

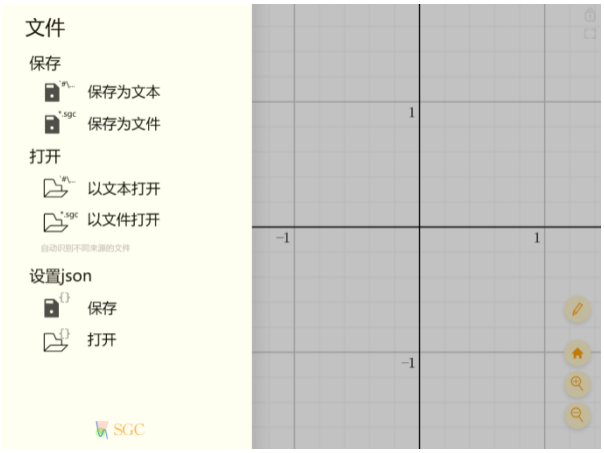
点击右上角“ ”锁定坐标系，这将使坐标系不能被移动、缩放，图标变为“ ”，右下方按钮变暗。再次点击这个图标即可解锁。

点击右上角“ ”进入全屏模式，再次点击即可退出。

1.2 菜单栏



菜单栏打开后有几个选项。点击文件打开文件面板。



“文件”中可以保存或者打开 sgc 文件。“文件”中还可以保存设置 json。

设置 json 将保存当前 SGC 的全部设置以及坐标系的位置和缩放倍数。这样，打开文件时，就可以恢复保存时坐标系的位置。

SGC json 遵循标准 json 格式，规范如下：

```
{
"SGCjson":"v0.2",
"fontsize" : 18, "highlight" : 1,
"sfspeed" : 0.5,
"mode" : 1,
"axis" : 1, "NumOfAxis" : 1, "grid" : 1,
"FPS" : 0, "Temp" : 0, "MousePos" : 0, "loadIcon" : 1,
"SpecialPoints" : 0, "dtrecalc" : 0, "InputNum" : 0,
"pqmin" : 0, "pqmax" : 6.283185307179586,
"pqprec" : 400, "pqminexp" : 0, "pqmaxexp" : "2π",
"polmin" : 0, "polmax" : 6.283185307179586,
"polprec" : 400, "polminexp" : 0, "polmaxexp" : "2π",
"dx" : 0 ,"dy" : 0, "sf" : 0
}
```

点击清屏将会弹出确认窗口,确认后将会清空屏幕,但是设置不会被还原。



点击检测会运行检测模式。若要进行检测，请先保存，否则会导致文件丢失。

检测模式使用一些 sgc 文件进行性能测试。检测时建议不要进行其他操作，以免导致 SGC 卡死或者闪退。检测结束后，将会有如下图的显示。因为检测模式会修改 SGC 的设置，建议检测完后重新启动 SGC。检测模式的基准分数是 1000。



点击切换模式可以切换到平面或者 3D 坐标系。



点击设置打开设置，点击关于打开关于页面。（**设置参见章节 10**）



点击高级功能可以选择打开动画模式以及脚本终端，点击相关的按钮即可。



点击素材集市打开素材库，参见 **1.4**。

1.3 键盘



打开键盘后,键盘上方会有一个橙色圆框,还有“123”（数字与符号）、“*f*(*x*)”（函数）、“*abc*”（字母）、“+”（外部导入）、“喇叭”（函数音频）几个按钮，点击前三个即可切换到对应的输入模式。

点击“+”从外部导入表达式。点击喇叭播放当前坐标系中函数的声音。“数字与符

号”模式可以输入 0～9 的数字、π、e、x、y、z 等常见量，也可以输入常用运算符,包括+,-,*,/, ^等。



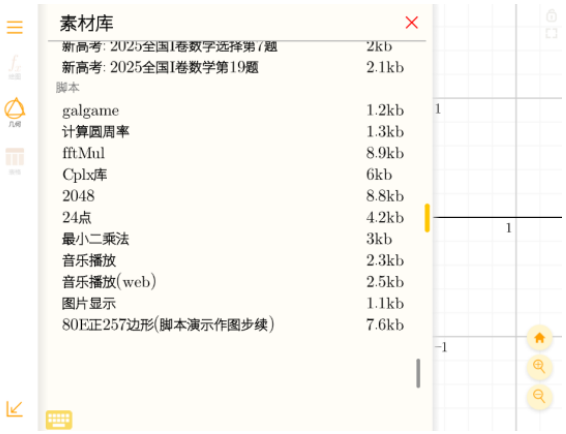
函数模式可以输入函数，例如三角函数、特殊函数。另外，点击“text”按钮可以插入一对引号(参见 **5.2**)。字母模式可以输入字母和一些常数。蓝色向上箭头按钮是 shift，点击后键盘左上角会出现蓝色“S↑”字样，某些按钮会变化。

长按键盘上的键可以连续输入。键盘右上角有一个红色小叉,点击可以关闭键盘。输入区被隐藏后将无法打开键盘。点击“删除”可以删除光标前一个字符，长按可以快速删除。类似箭头的按键是 enter,可以将你输入的内容写入输入区并绘制。在实际的键盘下按 enter 作用等同。

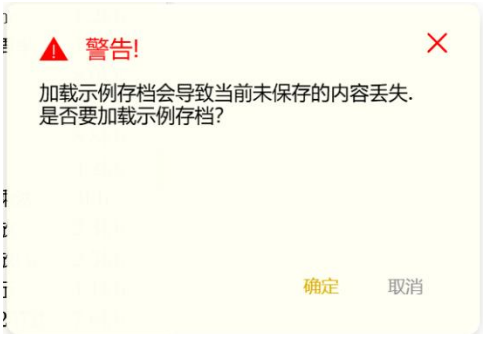
注意：错误的表达式在按下 enter 键后也会被输入。空白的内容不会被输入。

Shift 键的高级功能：按下 shift 再按下左或右按钮，可以快速跳到当前表达式开头或者结尾。

1.4 素材库

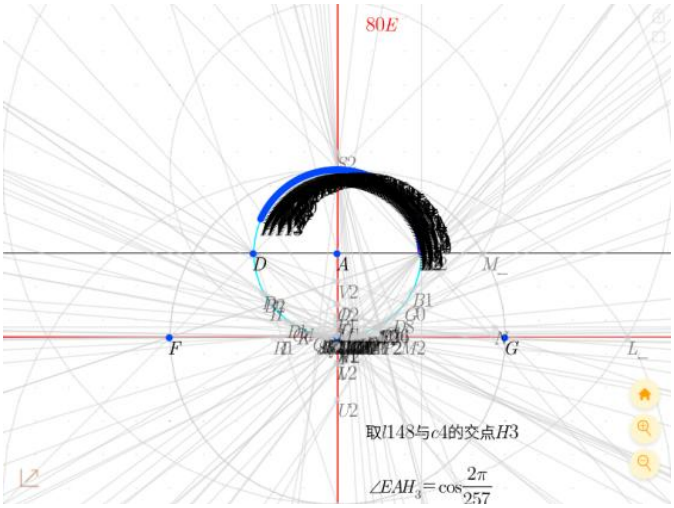


打开素材库后，出现如图所示界面。素材库分为几个部分：几何，函数，隐式，抽象，脚本等。其中，“脚本”中的素材，都需要运行脚本来实现效果。使用鼠标点击示例存档名称，即可打开这个存档。例如打开“80E 正 257 边形（脚本演示作图步续）”后，出现弹窗：



点击“确定”即可打开这存档。

打开菜单栏，选择“高级功能”中“脚本”，点击“运行”，得到：



（脚本没有运行完。省略脚本运行过程）

脚本的运行、更多信息参见**章节 8**

2 输入

2.1 输入方式

SGC 提供了多种输入方式。你可以在虚拟键盘从外部导入表达式，也可以打开虚拟键盘进行输入，还可以直接按真正的键盘输入。

如果您使用原版 Scratch 运行本软件，那么注意：由于 Scratch 3 不支持 backspace，故删除键用“\”键代替。长按可以快速删除。



点击确认即可在光标前插入你输入的内容。

SGC 的输入默认使用 CMU Serif 系字体，带有高亮显示，这样可以清晰地显示你输入的内容。函数是黄色，常量是绿色，运算符是蓝色，文本是灰色，其余内容是黑色。其中，自定义函数以斜体显示，系统函数以正体显示。

$$\sin(ax+b)$$

这高亮可以在设置中关闭，关闭后将显示

$$\sin(ax+b)$$

2.2 输入类型

在输入时，表达式左侧会有一个加号“+”。点击加号能打开输入类型小窗口，可以选择输入为“表达式”或者“纯文本”。纯文本可以作为注释使用，但是不会被计算。纯文本的语法错误会被忽略。

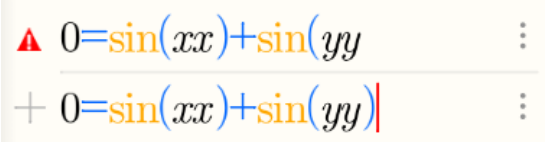


2.3 输入语法

SGC 将会自动检测输入内容，这称作“错误检查”,包括括号是否匹配、二元运算符

是否合理、是否缺少输入对象、输入是否有效等。

若你输入的内容有语法错误，左侧的加号“+”将变为一个红色“警告”符号“⚠”。如输入“0=sin(*xx*)+sin(*yy*”，将会显示警告图标，将光标移动到最后并输入“)”补全即可。输入完成后,按下 enter 即可。



中文符号无法解析，但是中文可以作为变量。

为了方便输入，SGC 拥有自动补全与错误检查的功能，能够很好地解析此类内容。

可以转换的内容如下：

输入	输入解释	转义
2sin(2*x*)	常量*函数 或 常量*变量	2·sin(2·*x*)
xx	变量*变量	*x*·*x*
*ax*sin(*x*)	变量*函数	*a*·*x*·sin(*x*)
*x*sin*x*	省略一个括号的函数	*x*·sin*x*
(*x*+1)(*x*-5)	省略乘号的两个式子	(*x*+1) ·(*x*-5)
2<*x*<5	中间部分只有一个内容的不等式	(2<*x*)and(*x*<5)

函数已经支持类似 sin*x* 的简写。但是必要的圆括号不可以省略。例如 sin(*x*/*y*)不可以简写为 sin*x*/*y*。

注意：遇到了可以解析成多种函数的内容，例如 asinh，会优先按照最长的函数进行解析。例如：想要输入 *a*·sin(*x*)，不可以输入 asin(*x*)，因为这会被解析成 arcsin(*x*)。

不可以转换的内容如下：

1. 2<(*x*+1)<5：中间部分包含两个及以上内容的不等式
2. sinsin*x*：两个不带括号的函数
3. 将 sin(*x*)^2 输入为 sin^2(*x*)

特别地，输入“pi”可以代替 π。

2.4 创建对象

SGC 功能丰富，可以绘制显函数、隐函数、参数方程、极坐标等多种类型的图像。

2.4.1 表达式计算

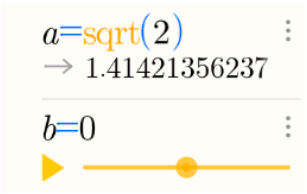
进行表达式计算，可以直接输入要计算的内容。如想要计算 1+1，那么在输入区输入“1+1”并计算，得到结果为 2。



2.4.2 变量

SGC 可以创建变量。变量分可播放变量和常量。例如创建变量 a，可以输入 $a=0$，也可以输入 a。SGC 会自动创建未定义变量并显示滑动条，但是这变量会被定义为可播放变量。常量不带有滑动条。

记变量内容为 u。如果 u 的表达式仅为一个实数，那么是可播放变量，否则是常量。比如 $a=\text{sqrt}(2)$是常量。若要创建 $a=1$ 这一常量，可以输入 $a=1+0$。

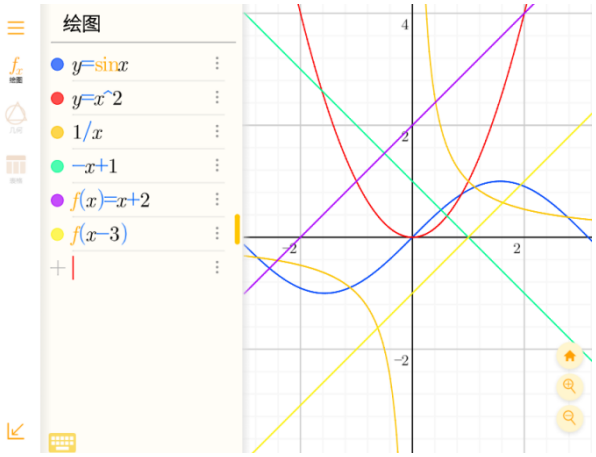


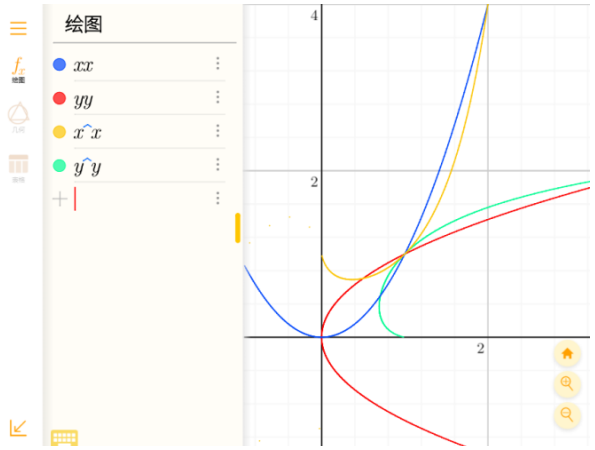
特别地，变量的表达式中不能具有 x、y、z。若要实现这样的变量，请使用函数定义。变量不能以数字开头，但变量名中可以带有 x、y、z。例如：myvar、good_name、$x1=0$、$y_2=0$ 是合法命名，1g、69lyc 是非法命名。若输入 69lyc=43，将会被解析为 $69l\cdot y\cdot c=43$，是一个隐函数。

定义了变量之后，后面的表达式都可以调用这个变量来计算。

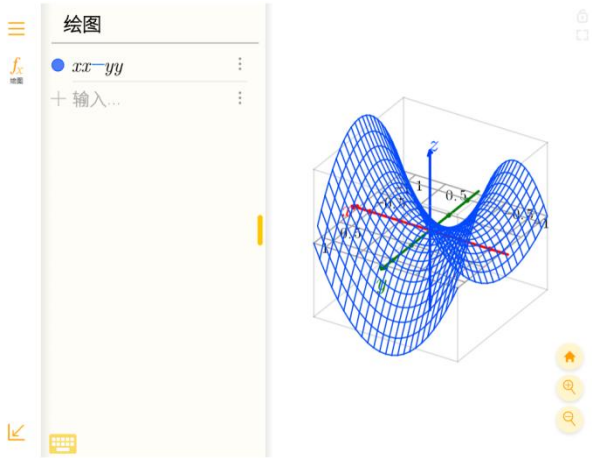
2.4.3 函数

创建一元显函数，只需直接输入函数的内容，“$y=$”不是必须的。如函数 $y=1+x$，可以直接键入 $1+x$，也可以输入 $y=1+x$ 或者 $f(x)=1+x$。以最后一种方式输入的函数，将可以在后面进行调用。SGC 同样也支持绘制 x 关于 y 的函数。输入 $x=f(y)$或者仅包含 y 的表达式即可，例如要绘制 $x=1+y$，可以输入 $x=1+y$ 或者 $1+y$、$f(y)=1+y$。





若要绘制三维显函数，切换到 3D 模式，输入一个既包含 x 也包含 y 的式子即可。例如要绘制曲面 $z=x^2-y^2$，输入“$xx-yy$”即可。

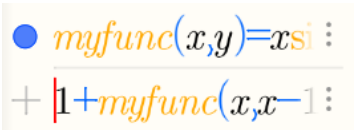


2.4.4 定义函数

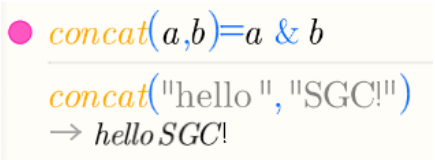
SGC 支持定义函数，并且支持定义拥有多个参数的函数。如定义函数 $f1$，有两个参数为 u，v，表达式为 $u-v\sin u$，那么输入 $f1(u,v)=u-v\sin u$。这时，计算输入，就可以成功定义这函数。

未定义的函数以黑色显示，已定义的函数以黄色斜体显示。

请注意：如果需要创建包含系统函数名的自定义函数，如 $mysin()$，需要在其中包含至少一个下划线，如 $my_sin(x)$为合法命名。定义 $myfunc(x,y)=x\sin y$，输入 $1+myfunc(x,x-1)$，显示如图：



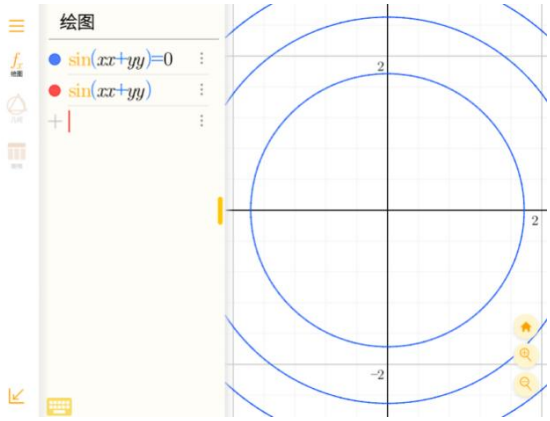
自定义函数还能以文本作为参数。如定义 concat(a,b)=a &b。计算 concat("hello ", "SGC!")，得到"hello SGC!"。



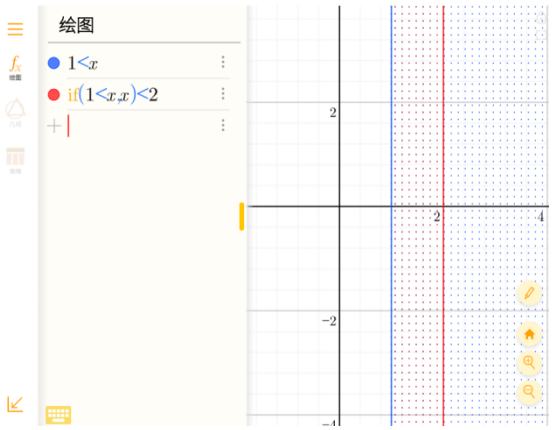
2.4.5 隐函数、不等式

绘制隐函数或者不等式，需要在表达式最外层有隐式（不等式）符号才能判定为隐函数（不等式）。如想要绘制 $\sin(xx+yy)=0$，输入“$\sin(xx+yy)=0$”，而不是“$\sin(xx+yy)$”或者“$(\sin(xx+yy)=0)$”。“$\sin(xx+yy)$”会被识别为二元显函数 $z=\sin(xx+yy)$并在 3D 模式下显示；“$(\sin(xx+yy)=0)$”则会被认为是布尔表达式，而不是隐函数。

如果隐函数包含 z，那么认为是一个三维隐函数（不等式）。



SGC 目前并不支持形如 $1<x<2$ 这样的不等式。如果要绘制这样的不等式，那么就需要利用 if。可以输入 $\text{if}(1<x,x)<2$ 代替 $1<x<2$。

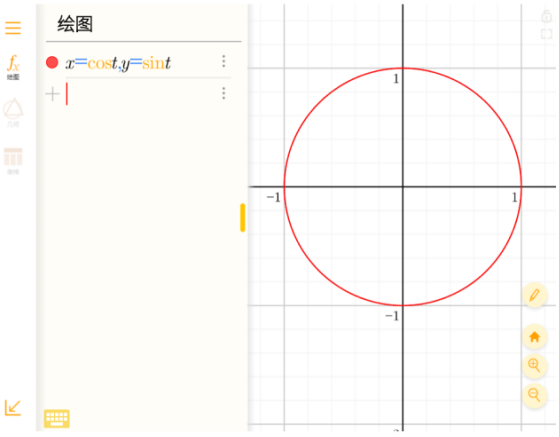


注意：SGC 使用 marching squares 算法，因此如$(x-y)(x+y)^2=0$、$(x-1)^2=0$ 这样包含切面为 $z=0$ 的隐函数可能无法准确绘制。

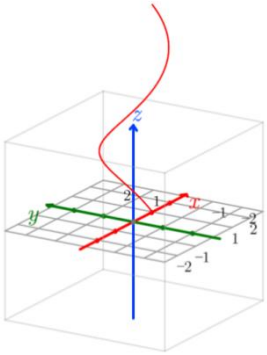
2.4.6 参数方程、极坐标

绘制二维参数方程的格式为：$x=f(t),y=g(t)$。

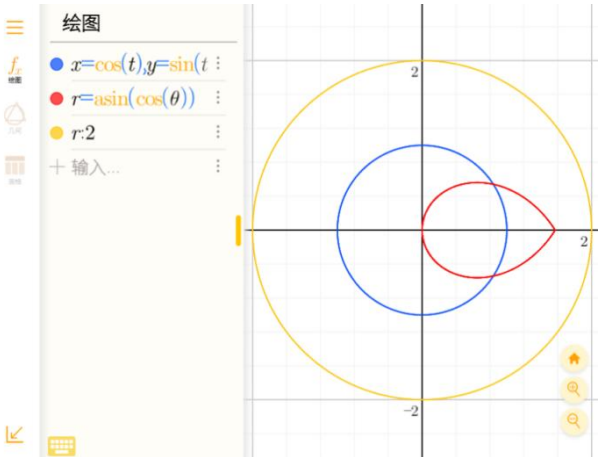
变量均为 t，t 的范围可以在设置中更改。如绘制半径为 1、圆心为(0,0)的圆，可以输入 $x=\cos t,y=\sin t$。



绘制三维参数方程的格式为：$x=f(t),y=g(t),z=h(t)$。例如 $x=\cos t,y=\sin t,z=t$：



若要输入极坐标函数，需要输入 $r=f(\theta)$或者 $r:f(\theta)$强行定义极坐标函数，避免和变量 r 冲突。例如绘制极坐标函数 asin(cos(θ))，输入 $r=$ asin(cos(θ))即可。



输入 $r:2$，这将强行绘制极坐标函数 $r=2$。

3 输入对象操作

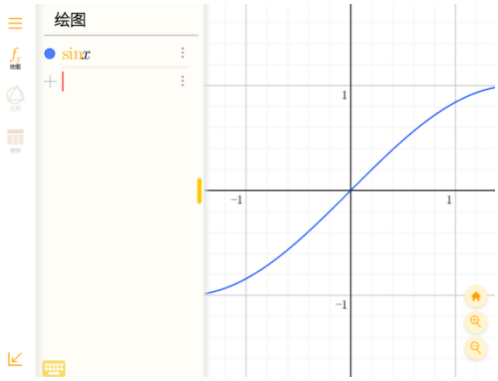
3.1 主界面操作

输入一个非常量、变量对象后,左侧会出现一个彩色圆形。常量、变量对象将不会有此圆形，而是空的。

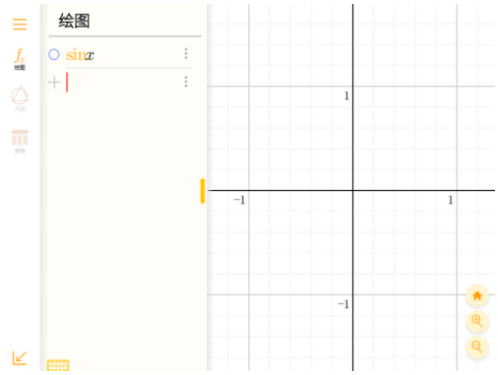
点击此圆形即可使当前对象显示或者隐藏。



1+1 的左侧没有圆形，但是 x 的左侧有。



输入了 sin$x$。现在显示的是 sin$x$ 的图像。



点击小圆形后，sin$x$ 这一对象被隐藏。

3.2 小窗口操作

一个对象的最右侧有一个 More 按钮（三个小点），如下图。



点击后将会打开小窗口,有若干选项。鼠标所指示的选项会被高亮，如下图。



对于不同类型的输入，More 菜单是不同的。对于函数等对象，如下图



对于有计算结果输出的，如下图的 1+1 输出结果为 2



点击“重复输入”将会把选择的对象复制到当前光标的位置之前。点击“重复输出”将会把选择的对象所得到的结果复制到当前光标的位置之前，如重复“1+1”的输出，会把“2”复制到当前光标的位置之前。

对于 v4.8.7 以下的版本，导出功能如下：

点击“导出”将会导出选择的对象至一个列表,三击列表显示的那一项进行复制。



对于 v4.8.7 及以上的版本，导出功能如下：

点击“导出”将会弹出一个窗口：



点击“导出当前表达式”即可复制当前的表达式。



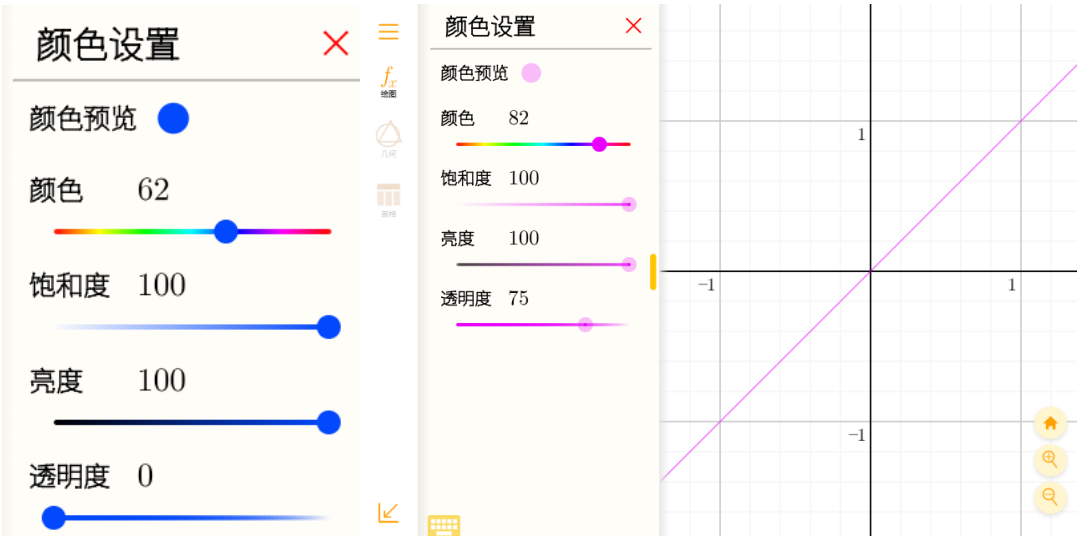
v4.8.7 及以上的版本有一个非常好的功能，可以把线性表达式转换为 LaTeX 表达式，点击“导出 TeX 表达式”即可。例如“1/x+1”使用这功能导出后，便有



如果表达式本身是有 LaTeX 标记的，那么将会返回 LaTeX 代码。

请注意：由于 SGC 大型运算符语法的原因，大型运算符不支持 LaTeX 语法，强制导出虽然不会产生错误，但是并不能显示成期待的样式。例如$\frac{\mathrm{d}}{\mathrm{d}x}(x^x)$的 SGC 线性表达式为 diff(x^x,x,x)。但是使用转换却得到\mathrm{diff}\left(x^{x},x,x\right)，而不是原本我们期待的的表达式。

点击“颜色”将打开样式设置面板。你可以更改颜色、饱和度、亮度和透明度，使用“颜色预览”可以预览修改后的颜色。



拖动滑动条即可修改参数. 四个参数最大值均为 100，最小值均为 0。

点击“删除”将会删除选择的对象。

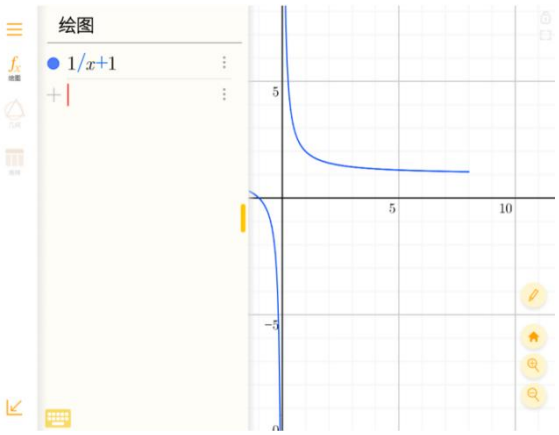
4 坐标系

4.1 平面坐标系

按住坐标系并拖动,可以改变坐标轴的位置。滚动鼠标滚轮，可以放大或缩小坐标系。滚动鼠标滚轮将会以鼠标为中心进行缩放，点击右侧小按钮将会以软件像素坐标系原点为中心缩放。

SGC 的显函数、隐函数采用区块加载，未加载的区域在拖动时不会被绘制，停止拖动后将会计算未加载部分并重新绘制。对于 v4.7 以后的版本，缩放坐标系后会延时计算，避免卡顿。

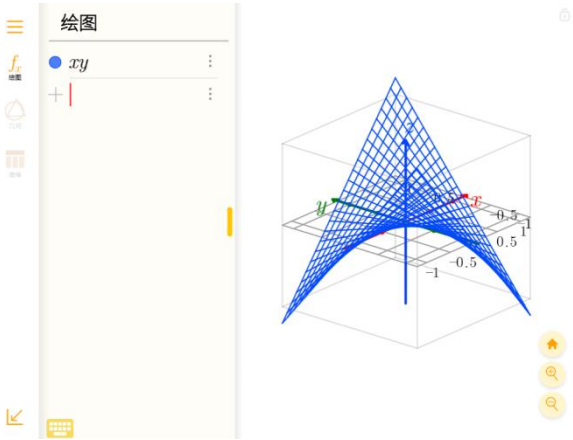
特别地，4.10 版本支持修改“延时刷新”选项，关闭这个选项后，将在拖动时每隔固定时间计算新的部分，但这可能会导致性能问题，因此这个选项默认开启。



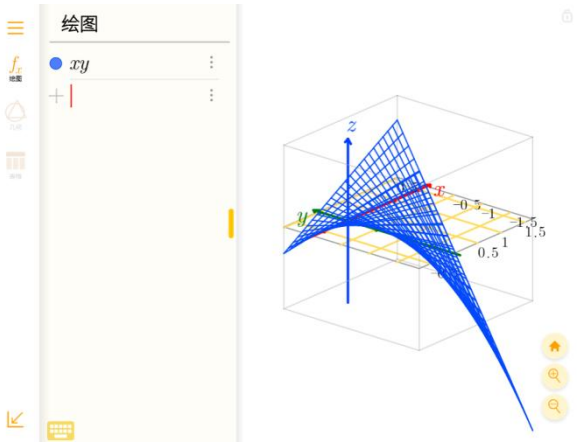
例如正在拖动坐标系，$1/x+1$ 尚未被计算的地方没有被绘制出来。

4.2 3D 坐标系

3D 绘制支持绘制二元显函数和 3D 方程、不等式以及 3D 参数方程。3D 坐标系 z 轴是固定的，x 轴和 y 轴可以转动。拖动 3D 网格时，网格会高亮。这代表正在平移 x-y 平面。使用鼠标滚轮进行缩放时，缩放中心是鼠标所指代的 x-y 平面上的一个点。



如图，这是默认 x-y 平面位置。



正在拖动 *x*-*y* 平面，网格被高亮。

5 计算引擎

SGC 采用最新计算引擎 v4，支持下列几种对象：

数(Number) （不区分 float 与 int）	1 -2.5 0.333333333333333
字符串(String)	"hello SGC!" "0.33333333333333" "SQY!+-*/"
布尔值(Boolean)	true false

5.1 技术参数

三角函数的参数范围:$-10^{10}<x<10^{10}$。对于所有函数，保留 15 位有效数字。

实数最大为 1.79769313486231e+308 即 2^{1024},最小为 1e-323。常数保留 15 位小数。

幂的计算：a^b 等价于 a^b。对于 a^b，若 $a>0$，返回 e^(b ln(a))；若 $a<0$，则试着将 b 化为分数：如果 b 的小数位数不足 7 位，认为 b 是一个有理数。如果 b 化为分数 x/y 后，x 或者 y 一者大于 10^8，认为 b 不是一个有理数。如果 b 是有理数，那么进行幂的运算。这样，SGC 能判断一些幂是否有意义：

```
(-1)^0.333
→ ?

(-1)^0.3333333333333333
→ -1

(-1)^(1/3)
→ -1
```

```
(−1)^(−0.1145)
→ ?
(−1)^(−0.114514)
→ ?
(−1)^(−0.1145141)
→ ?
(−1)^(−0.1145141919810)
→ ?
(−1)^(sin1)
→ ?
(−2^13)^0.07692307692307692
→ −2
```

另外，认为 0^0=1：

```
0^0
→ 1
```

5.2 计算

在计算中，任何 NaN 参与运算都得到 NaN。任何数除以 0 都得 NaN。

使用&来连接文本。如果 a&b 中，有任何一者不是数，那么会强行将这一者转换为一个字符串。例如：有一个变量 a=1，另一个变量 b=2。输入表达式 a&b 得到结果"12"。输入表达式"a="&a&",b="&b，得到结果"a=1,b=2"。

```
a=1
b=2
a&b
→ 12
"a="&a&",b="&b
→ a=1,b=2
```

5.3 运算符与函数

SGC 支持的运算符如下：

数学运算: +,-,*,/,^,!(阶乘)

字符串: &(字符串拼接，如"aa"&"bb"="aabb","a"&1="a1")

逻辑运算: <,>,=,<=,>=,==,&&,||

下表列举了 SGC 目前支持的函数。函数的输入不一定需要打括号,但用虚拟键盘输入的函数,都会带上括号。

函数表

(由于 scratch 不分大小写,这些函数也不需要区分大小写)

函数名	解释	注释
sin,cos,tan,csc,sec,cot	三角函数	弧度制
asin,acos,atan acsc,asec,acot	反三角函数	弧度制 arcsin 之类的也可以识别
sinh,cosh,tanh csch,sech,coth	双曲函数	
asinh,acosh,atanh acsch,asech,acoth	反双曲函数	
ln,log	对数函数	$\log(x)=\log_{10}(x)$
alog	反常用对数	$\text{alog}(x)=10^x$
abs	绝对值	
floor,ceil round	向下/上取整 四舍五入	
sqrt,cbrt	平方根,立方根	
min,max	最小值,最大值	min(*a*,*b*) max(*a*,*b*)
if ifelse	如果 如果否则	if(*x*>0,*x*+1) 不满足条件则返回 NaN ifelse(*x*>0,*x*,1-*x*)
case	选择	case(*u*,*x*,*v*,*y*) 如果 *u* 为真，则返回 *x* 并跳过 *v* 的计算，如果 *v* 为真，返回 *y* 但会先计算 *u*
gamma	伽马函数	
zeta	泽塔函数	
erf,erfc	高斯误差函数	
sgn	符号函数	参数>0 则返回 1,<0 返回-1,否则返回 0
lambertW	朗博 W 函数	$f(x)=xe^x$ 的反函数的主分支
expIntegral sinIntegral cosIntegral Li	指数积分 正弦积分 余弦积分 对数积分	

beta	beta 函数	beta(x,y)
psi	digamma 函数	
FresnelS FresnelC	菲涅尔正弦、余弦积分	
EllipticK	第一类完全椭圆积分	
EllipticE	第二类完全椭圆积分	
sinc	$\frac{\sin x}{x}$	
SinhIntegral CoshIntegral	双曲正弦、余弦积分	
not	非	not(x<0)
and or	与、或	(a)and(b)或者 (a)&&(b) (c)or(b)或者 (a)||(b)
mod	取模	mod(a,b)返回 a 除以 b 的余数 余数总是正的

特别地，4.10 及以上版本，ifelse(b,x,y)可以使用(b ? x : y)代替。例如 sgn 函数：

$$\operatorname{sgn} x = \begin{cases} 1, x > 0 \\ 0, x = 0 \\ -1, x < 0 \end{cases}$$

可以写为 sgn$x$ = (x==0 ? 0 : x/abs$x$)。

5.4 高级运算符

SGC v4.4 及以上版本支持了高级运算符。目前支持的高级运算符有 Sum, Prod, Int, Diff, fSolve。高级运算符支持嵌套。例如

$$\sum_{y=0}^{100}\sum_{x=0}^{100} 2x+y$$

输入 sum(sum(2x+y,x,0,100),y,0,100)：

```
sum(sum(2x+y,x,0,100),y,0,100)
→ 1530150
```

5.4.1 求和

求和的表达式为：Sum(f(v),v,min,max)。这是对 f(v)从 min 到 max 求和。这里的变量可以是任意的。上限、下限也可以是表达式，甚至是包含高级运算符的表达式。

例如：要计算

$$\sum_{x=1}^{100}\frac{1}{x^{10}}$$

输入 sum(1/x^10,x,1,100)。

```
sum(1/x^10,x,1,100)
→ 1.000994575128
```

这是 zeta(10)的一个很好的近似。

由于

$$\zeta(10)=\frac{\pi}{93555}$$

我们输入

```
(93555sum(1/x^10,x,1,100))^0.1
→ 3.14159265359
```

得到圆周率的前几位小数。

5.4.2 求乘积

求乘积的表达式为：Prod(*f*(*v*),*v*,min,max)。这是对 *f*(*v*)从 min 到 max 求乘积。

例如：要计算

$$\prod_{y=1}^{10}y$$

输入 prod(*y*,*y*,1,10)。

```
prod(y,y,1,10)
→ 3628800
```

这计算 10!=3628800。

特别地，对于某个 *v*，如果遇到了 *f*(*v*)=0，会立即停止这个乘积的计算。利用此性质，可以做大于 2 的质数判断。

注意：这里的 *n* 必须大于 2。这里我们使用朴素算法判断质数。要确定 *n* 是否是一个质数，我们可以遍历从 2 至$\lceil\sqrt{n}\rceil$的整数来试除。如果某大于等于 2 的整数 *v* 使得 mod(*n*,*v*)=0，那么这样的 *n* 就不是质数。

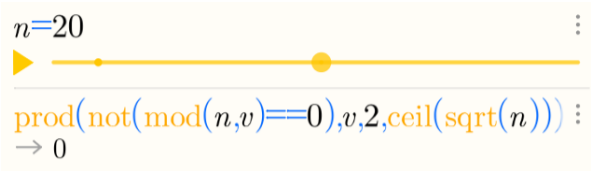
综上所述，我们总结出以下表达式：

$$\text{isPrime}(n)=\prod_{v=2}^{\lceil\sqrt{n}\rceil}\text{not}(\text{mod}(n,v)==0)$$

输入 prod(not(mod(*n*,*v*)==0),*v*,2,ceil(sqrt(*n*)))，并令 *n*=43，得到

```
n=43
prod(not(mod(n,k)==0),k,2,ceil(sqrt(n)))
→ 1
```

但是令 n=20，得到



不难得出，43 是质数，而 20 不是。

如果希望纳入 2 的判断，可以输入：

ifelse(n==2,true,prod(not(mod(n,v)==0),v,2,ceil(sqrt(n))))

或者是（n==2 ? true : prod(not(mod(n,v)==0),v,2,ceil(sqrt(n))))

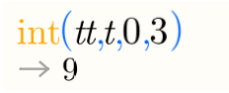
5.4.3 求积分

SGC 中，积分使用 Gauss-Kronrod 自适应数值积分。SGC 默认会用数值方法判断积分的敛散性，可以在设置中关闭。同样地，SGC 可以设置数值积分的递归上限。某些波动较大或者积分结果很大的函数，若得到了错误结果，可以适当调高递归上限。

求积分的表达式为：int(f(v),v,min,max)。这是对 $f(v)$从 min 到 max 求定积分。

例如：求$\int_0^3 t^2\mathrm{d}t$=9

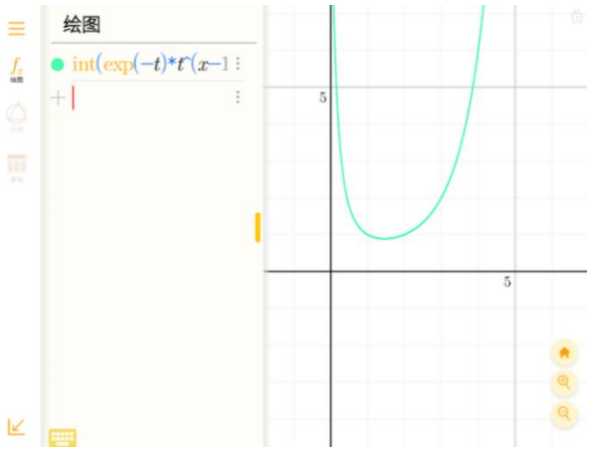
输入 int(tt,t,0,3)，得到



例如：gamma 函数的定义式

$$\mathrm{gamma}(x)=\int_0^{+\infty} \mathrm{e}^{-t}t^{x-1}\mathrm{d}t$$

由于取不到+∞，用 99 代替。输入 int(exp(-t)*t^(x-1),t,0,99)，得到函数



5.4.4 求导

求导的表达式为：diff($f(v)$,v, v_0)。这是 $f(v)$在 v=v_0 处求导。

求导使用中心差分：

$$\mathrm{diff}(f(x)) = \left.\frac{f(x+\Delta x) - f(x-\Delta x)}{2\Delta x}\right|_{\Delta x=10^{-5}}$$

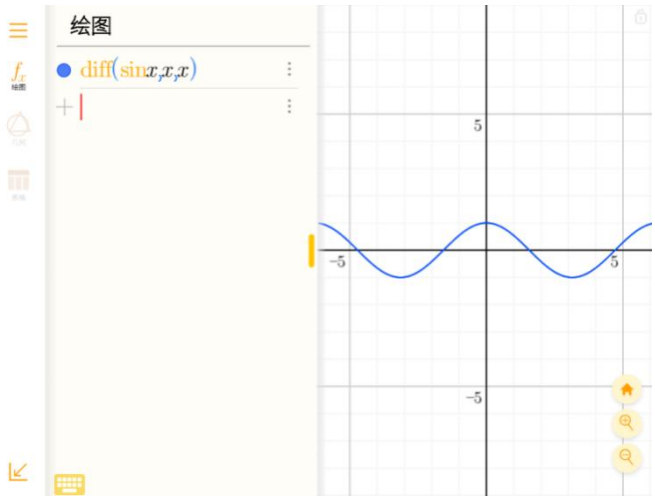
例如：要计算

$$\left.\frac{\partial}{\partial x}\sin(x)\right|_{x=1}$$

输入 diff(sin*x*,*x*,1)。

```
diff(sinx,x,1)
→ 0.540302305863
```

若要计算 sin(*x*)的导数，输入 diff(sin*x*,*x*,*x*)。



5.4.5 fSolve

这函数是使用 Newton 方法求数值解的函数。

例如 $f(x)=x^2-2$ 的零点为

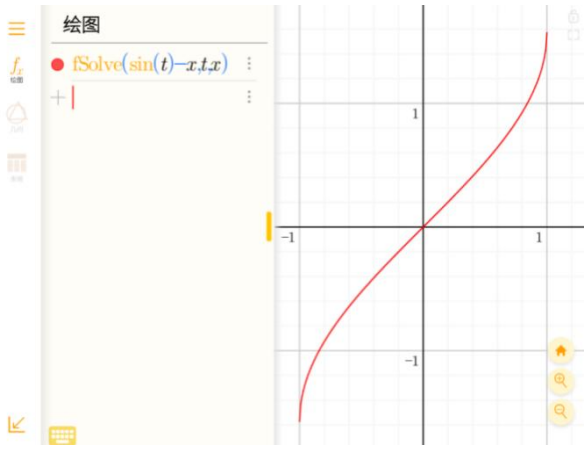
x:=[1.414213562373095, -1.414213562373095]，

想求这函数靠近 1 的零点，输入 fSolve(x^2-2,*x*,1)，获得输出为 1.414213562373095.

```
fSolve(x^2−2,x,1)
→ 1.414213562373
```

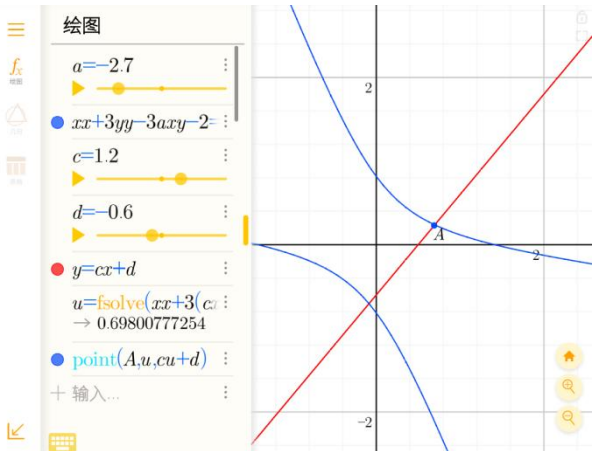
例如函数 *g*=arcsin *x* 是 *f*=sin *x* 的反函数，这也可以用 fSolve 描述。

这实际上在求解关于 *t* 方程 sin *t*=*x*。输入 fSolve(sin(*t*)-*x*,*t*,*x*)得到图像



同样地，fSolve 可用于圆锥曲线交点以及任意函数交点的探究。

例如：想要探究圆锥曲线 $xx+3yy-3axy=2$ 与 $y=cx+d$ 的交点问题，若需要获取靠右的一个交点，可以联立两方程，用 u=fSolve($xx+3(cx+d)^2-3ax(cx+d)-2,x,1$)来确定交点横坐标。如图中 A 点即为待求交点。



5.5 构造指引

利用上述的函数，可以构造很多方便实用的函数。

5.5.1 分段函数

在 SGC 中，有多种方式构造一个分段函数。

例如，要构造 $y=\begin{cases}-x, & x<0\\ x^2, & x\geq 0\end{cases}$，可以输入：ifelse($x$<0,-$x$,$x$^2)，这是用 ifelse 判断。同样，也可以利用 SGC 的逻辑算符：(x<0)(-x)+(x>=0)x^2。当 x<0，此表达式相当于 (1)·(-x)+(0)·x^2，反之同理。

5.5.2 距离

若要获取一个点(x,y)到原点的距离，可以定义函数：dis(x,y)=sqrt($xx+yy$)。

类似地，若要获取一个点(x_1,y_1)到点(x_2,y_2)的距离，可以定义函数：dis2(x_1,y_1,x_2,y_2)=sqrt((x_1-x_2)^2+(y_1-y_2)^2)。

5.5.3 定义域的限制

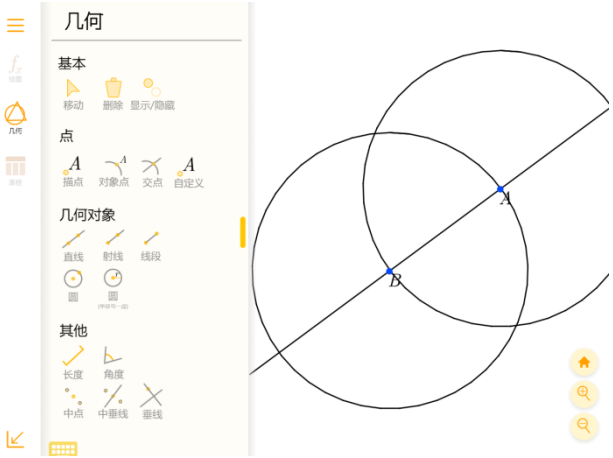
若要限制函数定义域，可以使用 if 函数。例如，想要绘制$-1\leq x\leq 2$的 $y=x^2$，可以输入：y=if(-1<=x<=2,x^2)。

如果使用 y=(-1<=x<=2)x^2，效果会有不同。因为在不满足条件时，(-1<=x<=2)=0，于是 y=0；而使用 if，不满足条件会返回 NaN，不会画出这区间上的函数。

限制定义域也可以使用定义域有限的函数来实现。由于 SGC 没有 CAS 功能，表达式不会被简化，类似 x+0 的式子在内部依然使用 x+0 来计算。因此，上述式子可以用 y=x^2+0/sqrt(1+x)+0/sqrt(2-x)来代替。

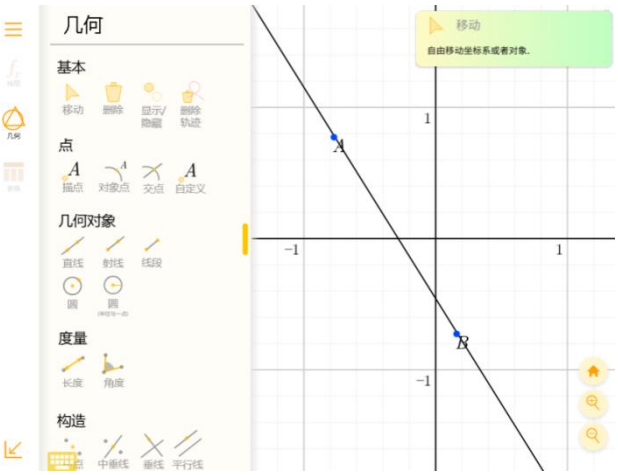
6 几何

SGC 提供了易用的几何功能,可以绘制点、线、圆、文本、多边形等对象。

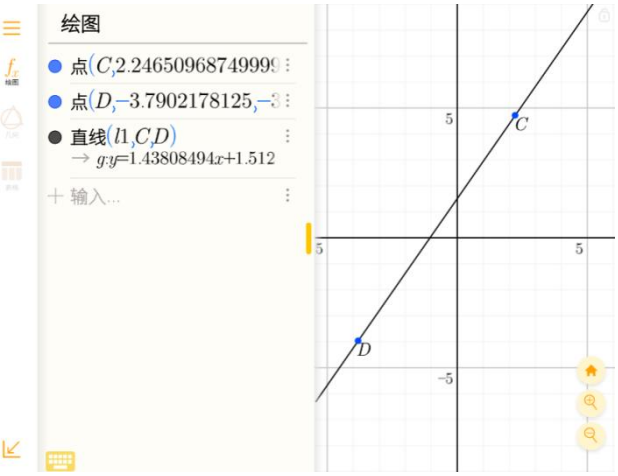


6.1 基本操作

左边模式栏选择“几何”，出现如图所示的几何页面。使用鼠标拖动或者鼠标滚轮可以移动几何功能页面。点击一个几何功能，右侧会弹出提示。您可以根据提示完成操作。



点击“绘图”，几何内容会在输入区显示。



在几何内容中，直线的解析式会显示出来，并且以 g:开头。

几何支持的功能如下：

函数名	解释
点(名称,*x*,*y*)	在(*x*,*y*)处创建点
对象点(名称,所在对象[,因子])	在对象上创建一个点；如果填入了因子，那么对象点的位置由这个因子确定
交点(对象 1,对象 2,名称 1,名称 2)	确定两个对象的交点。如果有两个，那么两个交点均被创建；不需要的交点使用?填充
直线(名称,点 1,点 2) 射线(名称,点 1,点 2) 线段(名称,点 1,点 2)	创建过点 1 和点 2 的线
圆(名称,圆心,圆心外一点)	根据两个点创建一个圆
圆规(名称,圆心,点 1,点 2)	创建一个圆，但半径为点 1 到点 2 的距离
距离(变量名,点 1,点 2)	测量点 1 到点 2 的距离并赋值给变量
角度(变量名,点 1,点 2,点 3)	测量点 1-点 2-点 3 所成的角的度数，并赋值给变量
中点(名称，点 1，点 2)	创建两点的中点
中垂线(名称，点 1，点 2)	创建过两点中点的垂线，使得该直线上任意一点到这两点的距离均相等
垂线(名称，直线 1，点 1)	过点 1 做直线 1 的垂线
平行线(名称，直线 1，点 1)	过点 1 做直线 1 的平行线
旋转(旋转中心,旋转对象,名称,旋转角)	将旋转对象绕着旋转中心旋转一定角度
对称(对称轴,对称对象,名称)	将对称对象沿着对称轴做对称
是否垂直(直线 1，直线 2)	检验两直线是否垂直
是否平行(直线 1，直线 2)	检验两直线是否平行
多边形(poly，点 1,点 2, ... ,点 n)	创建名称为 poly 的多边形
面积(变量名，对象)	测量多边形或者圆的面积
角平分线(名称,点 1,点 2,点 3 [,等分数 n])	做角的(*n*)等分线，点 2 是顶点。*n* 默认是 2。

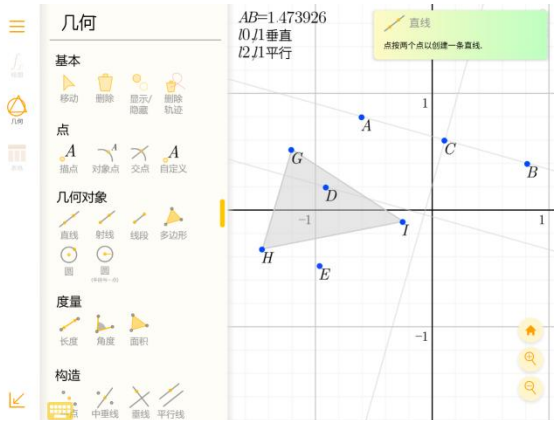
在 4.7.3 以上版本中，几何函数支持了英文：

函数名	英文名
点(名称,*x*,*y*)	point(名称,*x*,*y*)
对象点(名称,所在对象[,因子])	objpoint(名称,所在对象[,因子])
交点(对象 1,对象 2,名称 1,名称 2)	inter(对象 1,对象 2,名称 1,名称 2)
直线(名称,点 1,点 2) 射线(名称,点 1,点 2) 线段(名称,点 1,点 2)	line(名称,点 1,点 2) ray(名称,点 1,点 2) segment(名称,点 1,点 2)
圆(名称,圆心,圆心外一点)	circle(名称,圆心,圆心外一点)
圆规(名称,圆心,点 1,点 2)	compass(名称,圆心,点 1,点 2)
距离(变量名,点 1,点 2)	distance(变量名,点 1,点 2)
角度(变量名,点 1,点 2,点 3)	angle(变量名,点 1,点 2,点 3)
中点(名称，点 1，点 2)	midpoint(名称，点 1，点 2)
中垂线(名称，点 1，点 2)	perpbisector(名称，点 1，点 2)
垂线(名称，直线 1，点 1)	perpline(名称，直线 1，点 1)
平行线(名称，直线 1，点 1)	paraline(名称，直线 1，点 1)
旋转(旋转中心,旋转对象,名称,旋转角)	rotate(旋转中心,旋转对象,名称,旋转角)
对称(对称轴,对称对象,名称)	reflect(对称轴,对称对象,名称)
是否垂直(直线 1，直线 2)	isperp(直线 1，直线 2)
是否平行(直线 1，直线 2)	isparallel(直线 1，直线 2)
多边形(poly，点 1,点 2, … ,点 n)	polygon_g(poly，点 1,点 2, … ,点 *n*)
面积(变量名，对象)	area(变量名，对象)
角平分线(名称,点 1,点 2,点 3 [,等分数 n])	anglebisector(名称,点 1,点 2,点 3 [,等分数 *n*])

对象点的因子可以是一个表达式。特别地，对象点还可以创建函数上一点。例如要创建 $y=\sin x$ 上的一个点 A，既可以使用 point(A,t,sin$t$)，也可以使用 objpoint(A,sin$x$)。但是这两者有所不同。前者不可以被拖动，而后者则可以被拖动。使用 objpoint(A,sin$y$) 则可以创建 $x=\sin y$ 上的一个点 A。如果使用 objpoint(A,sin$x$)，那么不可以填入第三个参数。请使用 point 代替。

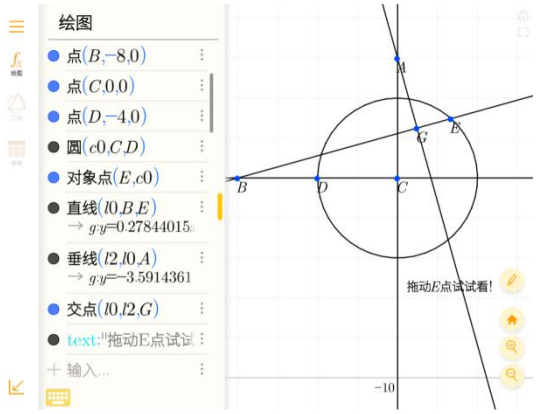
6.2 创建几何对象

创建对象时，非当前选择类型的对象以灰色显示。例如选择“直线”工具，需要选择两个点。这时，坐标系上所有不是点的几何对象都将变为灰色，便于提示选择的对象。

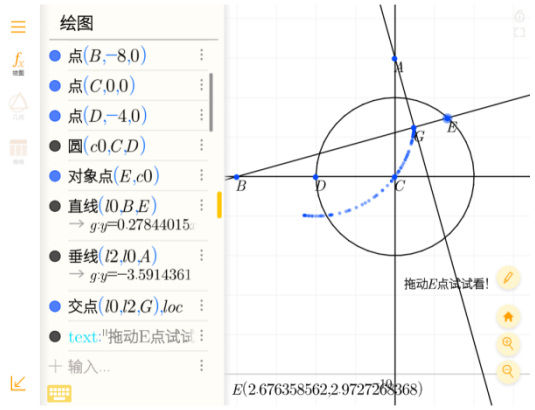


6.3 轨迹

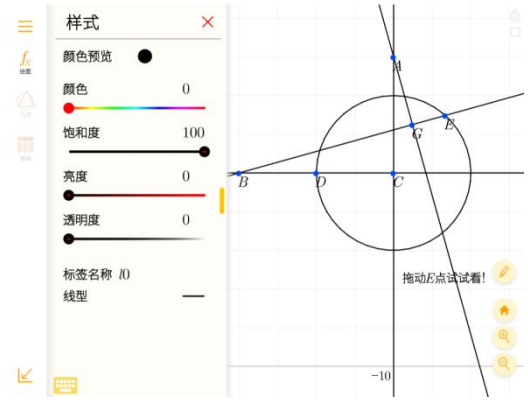
若要显示某个点的轨迹，则在表达式的最末尾添加“loc”。例如下面这个例子：



要显示 G 的轨迹，使用“交点($l0$,$l2$,G),loc”。拖动 E 点后得到



6.4 修改对象样式



打开样式设置面板后，可以在颜色面板下方修改线型为实线或者虚线。

在 4.10 以上的版本中，还可以修改几何对象的标签。如果某个点的标签为空，那么显示这个点的名称，否则显示标签内容。在样式设置面板中，点击标签名称即可修改。标签内容支持高级文本样式，参见 **7.2**。

7 文本

7.1 普通文本

文本只能在平面坐标系中显示。在坐标系中显示文本，使用以下格式：

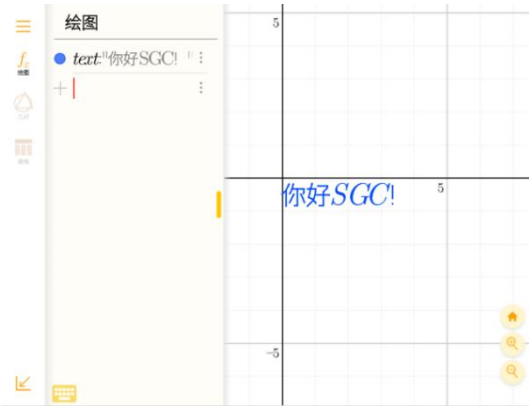
text:<text>,*x*,*y*,size,type

其中,type 是 0 或者 1。0 表示字号大小和坐标系缩放相关，1 则表示不相关。

例如：想要在坐标系(0,0)处以“1”号字显示文本："你好 SGC!"，那么输入：

text:"你好 SGC!",0,0,1,0

效果如图：



7.2 高级文本

高级文本在几何文本中起作用。高级文本支持转义符，转义符可以嵌套。

下表中转义符使用红色标记。

/r Hello! /-r	Hello!	用 regular 字形渲染
/b Hello! /-b	***Hello!***	加粗字体
/s3 Hello! /-s3	*Hello!*	放大 3 倍
/u Hello! /-u	*Hello!*	下划线
He/sullo! /su	$He^{llo!}$	上标
He/sdllo! /sd	$He_{llo!}$	下标

放大的倍数是 1、2、3、4、5、6、7、8、9、a(10)、b(11)、c(12)、d(13)、e(14)、f(15)中的一种。

例如想要显示“H_2O”，使用高级文本“/rH/sd2/-sdO/-r”：

text_g(Gtext0,"/rH/sd2/-sdO/-r",-8,-6)

H_2O

7.3 STeX

FastSTeX − LaTeX anywhere$^{\in \text{Scratch}}$

SGC 中有一个 FastSTeX v1.2 终端，能够渲染一些 LaTeX 公式，并储存编译后的表达式以提高渲染性能。FastSTeX 的大部分语法遵循 TeX 语法规则。

TeX 文本大小不会随坐标系缩放改变。

`\frac{a}{b}`	$\frac{a}{b}$
`\sqrt{a}`	$\sqrt{a}$
`\sqrt[k]{x}`	$\sqrt[k]{x}$
`\overset{v}{u}`	u 上的 v
`\underset{v}{u}`	u 下的 v
`a^{b}`	a^{b}
`a_{b}`	a_{b}
`a^{x}_{y}`	a^{x}_{y}
`\overrightarrow{AB}`	$\overrightarrow{AB}$
`\overline{x}`	$\overline{x}$

`\int`	积分号
`\infty`	无穷大符号
`\sum`	求和符号
`\prod`	求积符号
`\partial`	偏导符号
`\pi`	π
`\longeq`	长等号(==)
`\+空格`	空格
`\mathrm`	正体
`\lfloor`	左侧 floor
`\rfloor`	右侧 floor
`\lceil`	左侧 ceil

`\left{`	$\left\{$
`\right{`	$\right\}$
`\left[`	$\left[$
`\right[`	$\right]$
`\left|`	左侧绝对值符号
`\right|`	右侧绝对值符号

支持一些连字：

`=>`	\Rightarrow
`->`	\rightarrow
`==`	\longeq

STeX 具有一个非常好的功能，是 LaTeX 所不具备的。STeX 中的括号如果匹配，会自动变成合适的高度，不需要输入\left，\right。例如想要创建这公式

$$\left(x+\frac{1}{x}\right)$$

使用 LaTeX，则需要 \left(x+\frac{1}{x} \right)；若使用 STeX，只需(x+\frac{1}{x})。

另外一个非常方便的功能，是隐式补全括号。如果输入了以下式子：

$$\sqrt{\left(1+\frac{1}{x}$$

不难发现这式子缺少了一个右括号。使用 LaTeX，则需要\sqrt{\left(1+\frac{1}{x} \right.}；若使用 STeX，只需输入\sqrt{(1+\frac{1}{x}}。

如果要在坐标系中显示 TeX 文本，可以输入：latex:<latex>,*x*,*y*,size,0。或仍使用 text 标签：text:<latex>,*x*,*y*,size,t，这将会以 TeX 解析文本。*x*、*y*、size 可以是表达式，这便能创建可以“动”的公式。如变量 a=2.5，在(0,0)处以 0.6 点 TeX 字号显示公式

$$\frac{1}{\text{<变量 }a\text{>}}=\text{<}1/a\text{ 的值>}$$

输入：latex:"\frac{1}{"&a&"}="&1/a,0,0,0.6。这会先计算文本表达式"\frac{1}{"&a&"}="并获得输出，然后调用 TeX 解析结果。例如 a=2.5 时，字符串"\frac{1}{"&a&"}="的结果为"\frac{1}{2.5}"，再使用 TeX 进行解析、渲染字符串"\frac{1}{2.5}"。

$$\frac{1}{2.5}=0.4$$

拖动滑动条能看到实时刷新的 TeX 公式。

在(0,0)处以 0.6 点 TeX 字号显示公式

$$e^{i\pi}+1=0$$

$$e^{i\pi}+1=0$$

下面是一些例子：

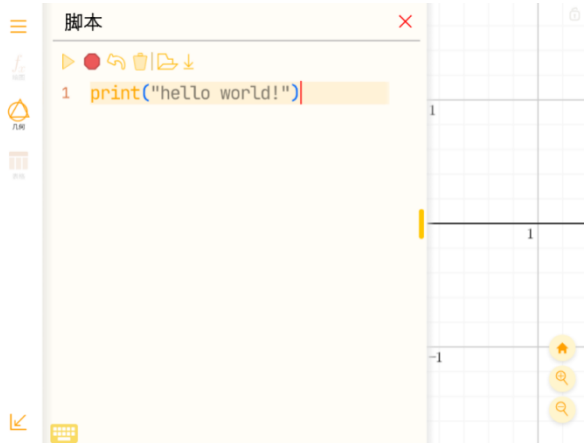
显示	STeX 代码	LaTeX 代码
$\int_0^1 \frac{1}{1+x^3}\mathrm{d}x=\frac{2\sqrt{3}\pi+\ln 64}{18}$	\int^{1}_{0}\frac{1}{1+x^3} \mathrm{d}x= \frac{2 \sqrt{3} \pi+ln64}{18}	\int^{1}_{0}\frac{1}{1+x^3} \mathrm{d} x=\frac{2\sqrt{3}\pi+\ln64} {18}
$f(x)=a_0+\sum_{n=1}^{\infty}\left(a_n\cos\frac{n\pi x}{L}+b_n\sin\frac{n\pi x}{L}\right)$	f(x)=a_0+\overset{\infty} {\underset{n=1}{\sum}}(a_ n cos \frac{n \pi x}{L}+b_n sin \frac{n \pi x}{L})	f(x)=a_0+\sum^{\infty}_{n=1 }\left(a_{n} \cos \frac{n \pi x}{L}+b_{n} \sin \frac{n \pi x}{L} \right)
$\sqrt{x_1^2+x_2^2}$	\sqrt{x^{2}_{1}+x^{2}_{2} }	\sqrt{x^{2}_{1}+x^{2}_{2}}
$\frac{\partial x^x}{\partial x}=x^x(1+\ln x)$	\frac{\partial x^x}{\partial x}=x^{x}(1+\ln x)	\frac{\partial x^x} {\partial x} =x^{x} \left(1+\ln x\right)

8 脚本

SGC 搭载简单的 SGC Script 语言。SGC Script 是一个栈式编译语言，能操作 SGC 底层，修改 SGC 设置，做出简单易懂的演示动画。SGC Script 已更新至 SGC Script II。

8.1 脚本编辑器

在高级功能中选择进入编辑器，即可打开脚本编辑器.



(示例 1)

脚本编辑器采用 JetBrains Mono 字体，支持一些连字。例如

```
a=0
while a⩽100{
    if a≠15 || a≠26{
        print(a)
    }
}
```

您可以上下左右移动光标，按键盘或者虚拟键盘插入字符，按下 enter 新建一行。当前光标所在的那一行会以不同颜色标注。

操作区按钮如下：

v4.8 以下的版本，操作区按钮如图：



v4.8 及以上的版本，操作区按钮如图：



按下第一个图标运行您的脚本，按下第二个图标强行停止您的脚本，按下第三个图标返回脚本编辑器，按下第四个图标清空脚本。

按下第五个图标打开脚本，第六个图标保存脚本。

例如运行示例 1 代码，显示



在脚本编辑器中，文字会高亮。注释为灰色，其他的高亮和函数一样。SGC Script 关键字为粉色。

8.2 语法

使用//作为注释开始的标记，这之后，当前行的所有内容都将被认为是注释。

8.2.1 数据类型与运算符

支持的数据类型为：

数(number) （不区分 float 与 int）	1 -2.5 0.333333333333333
字符串(String)	"hello SGC!" "0.33333333333333" "SQY!+-*/"
布尔值(Boolean)	true false
数组/列表(Array)	[1, 2, 3] [1, [2, [3]]] []

支持的运算符为：

+, -, *, /, ^ ,<, >, ==, <=, >=, !=, %, &&, ||, and, or, not

数学函数为：

sin, cos, tan, ln, log, sqrt, abs, floor, ceil, round, min, max, random

字符串相关函数：substr, length,string, number, find, count, concat

列表相关函数：length, flatten, find, count, concat

扩展函数：load, get, getFileUrl, getInput, ticks

函数不区分大小写。接收固定个数参数的函数，属于 BUILTIN 类型。能接收不固定个数参数的函数如 min, max，属于 BUILTIN_VAR 类型。

关键字还有：split, define, local。

部分函数说明如下：

函数名	解释
substr(string,*a*,*b*)	字符串 string 的第 *a* 到 *b* 个字符
length(*a*)	对于列表 *a*，返回 *a* 的长度 对于字符串 *a*，返回 *a* 的字符数 对于数 *a*，返回数化为字符串之后的字符数
find(*a*,*b*)	对于列表 *a*，寻找 *a* 中 *b* 第一次出现的位置 对于字符串 *a*，寻找字符串 *a* 中 *b* 第一次出现的位置；如果没有找到，返回 0
count(*a*,*b*)	寻找 *a* 中 *b* 出现的次数
load() getFileUrl()	从本地加载文件作为字符串 从本地加载文件作为 URL
input(title)	从外部输入字符串，会询问 title 中的内容
get(url)	获取 url 的内容
round(*a*[,b])	将 *a* 保留 *b* 位小数；若未给出 *b*，则返回四舍五入 *a*
random(*a*,*b*)	在 *a* 到 *b* 的范围内取随机数
ticks()	返回计时器值(ms)

函数参数个数不匹配将会抛出错误，如：

```
get()
```

```
·> Error: Function ‘get’ expected 1 argument, but 0 were given
```

8.2.2 变量

SGC Script 目前支持全局变量和局部变量。

变量的第一个字符不能是数字。不能用**关键字**命名变量。

变量可以放开头声明：

```
a;
b;
```

可以显式赋值：

```
a = 1;
b = "text";
```

局部变量需使用 local 声明，如在函数中。参见 **8.2.6**。

8.2.3 列表

列表索引从 1 开始。定义列表可以使用字面量：

```
c = [0,1,2,3,4,5];
d = [];
```

也可以使用 Array 声明：

```
c = Array[5];
```

d 是一个空列表。支持嵌套列表，即列表的元素是列表：

```
arr = [0,1,2,[3,4,5]];
d = [];
```

列表支持负索引，例如 *c*[-1]代表 *c* 的倒数第一项。

另外，这种表示方法还可以用来获取字符串指定位置上的字符。如：

```
print("hello SGC!"[1]);
print("hello SGC!"[-1]);
```

```
·> "h"
·> "h!"
```

"hello SGC!"[1]表示这个字符串的第一个字符，返回"h"；"hello SGC!"[-1]表示这个字符串的最后一个字符，返回"!"。

列表中的元素可以被替换：

```
c = [0,1,2,3,4,5];
c[2] = "aaa";    //替换列表 c 第二项为"aaa"
```

如果正索引超出了列表的长度，并且索引是一维的，将会自动填充：

```
test = [0,1,2] ;
test[5] = 3;
```

这时列表 test 将变为[0,1,2,3]。

```
test = [0,1,[0,1,2]] ;
test[2][5] = 3;
```

这时将抛出错误：

```
·> Error: Array index out of range
```

如果负索引超出了列表的长度，如：

```
test = [0,1,2];
print(test[-5]);
```

将抛出错误：

```
·> Error: Array index out of range
```

使用 append 在列表后添加元素。

```
test = [0,1,2];
test.append(3);
```

使用 insert 在列表指定项前添加元素。

```
test = [0,1,2];
test.insert(2, 666);
```

如果索引超出了列表的长度，将抛出错误：

```
·> Error: Array index out of range
```

使用 remove 删除列表指定元素。

```
test = [0,1,2];
test.remove(2);
```

如果索引超出了列表的长度，将抛出错误：

```
·> Error: Array index out of range
```

列表具有 split 方法，可以按照指定的字符进行拆分。

```
c = [];
print(c.split("a,b,c,SGC"));
```

```
·> ["a","b","c","SGC"]
```

默认用","作为分隔符。若要使用其他分隔符，例如"|"：

```
c = [];
print(c.split("a|b|c|SGC","|"));
```

```
·> ["a", "b", "c", "SGC"]
```

使用 flatten 压平列表：

```
print(flatten([0,1,[0,1,[0,1,2]]]))
```

```
·> [0, 1, 0, 1, 0, 1, 2]
```

使用 concat 串联多个列表：

```
print(concat([0,1],[2],"SGC"))
```

```
·> [0, 1, 2,"SGC"]
```

8.2.4 循环

SGC Script 目前只支持 while 循环。其语法如下：

```
a = 0;
while a <= 100 { a++; }
```

使用 break 能打断当前循环。

```
a = 0;
while a <= 100{
    a++;
    if a == 50{
        break;
    }
}
print(a)
```

循环支持嵌套。

```
a = 0;
while a <= 10{
     a++;
     b = 0;
     while b <= 10{
          print([a, b]);
          b++;
     }
}
```

8.2.5 如果

如果语法如下：

```
k = input();
if k % 2 == 0{
    print(k + "是偶数");
}
```

如果否则语法如下：

```
j = input();
if j % 2 == 0{
    print(j + "是偶数");
} else {
    print(j + "是奇数");
}
```

8.2.6 函数

SGC Script 可以定义函数。定义函数的语法如下：

```
define hello(){
    print("hello sgc!");
}
```

函数可以具有返回值。

```
define hello(x){
    print(x);
    return x;
}
```

函数支持递归，如使用递归方法计算 Fibonacci 数列。

```
define Fib(x){
    if x == 1 || x == 2 {
        return 1;
    } else {
        return Fib(x - 1) + Fib(x - 2);
    }
}
print(Fib(6));
```

```
·> 8
```

若需声明局部变量，使用 local。如：

```
define f(x){
    if x > 2 {
        return;
    }
    i = 0;
    while i < 3 {
        print([i, x]);
        f(i);
        i++;
    }
}
```

与：

```
define f(x){
    if x > 2 {
        return;
    }
    local i = 0;
    while i < 3 {
        print([i, x]);
        f(i);
        i++;
    }
}
```

两段代码输出不同。

8.3 操作 SGC 底层

8.3.1 输入操作

您可以使用 SGC Script 操作 SGC 的底层。这需要使用 system 库。这包括：新建输入、移除输入、修改输入颜色。

新建输入可以在当前输入区最后添加一个输入。例如：

```
addinput("y=x^2");
```

这将添加函数 $y=x^2$。注意：addinput 接收的参数必须是一个字符串，如果传入数，那么会将数转换为字符串。

如果想指定添加输入的颜色、是否显示，那么：

```
addinput("y=x^2", "00999900", 1);
```

这会指定添加的函数颜色为红色。

删除指定的输入，可以使用：

```
delinput(3);
```

这将删除第 3 个输入。如果不传入参数，那么将会清除所有的输入。

如果想设置输入某一项的颜色为 a，可以使用：

```
a = "62999900";
sgcInputColor(2, a);
```

这段代码会把输入的第 2 项颜色设置为蓝色。

8.3.2 图形操作

SGC Script 能控制 SGC 的默认坐标系，包括控制位置、缩放大小。一般地，坐标系默认缩放大小为 100，位置为(0,0)。调整的是坐标系的偏移位置。

要移动坐标系，使用 sgcTrans：

```
sgcTrans(1.5, 1);
```

这会把坐标系原点向左移动 1.5 个单位，向上移动 1 个单位。

```
sgcScale(sgcScale() / 2.6);
```

这会把单位长度设为原来的 2.6 分之一。

使用 sgcTrans()[1]、sgcTrans()[2]获取当前 x、y 轴正向偏移量。使用 sgcScale()获取当前缩放值。

8.4 系统常数

使用 sgcInput()获取所有输入。

使用 ticks()获取自程序启动后的计时器值。例如：

```
time_stamp = ticks();
define reset(){
    time_stamp = ticks();
}
define getsec(){
    return (ticks() - time_stamp) / 1000;
}
```

使用 time()获取 24 小时制时间。

```
define fix(x){
    return ifelse(length(x) == 1, "0" + x, x);
}
define myClock(){
    return fix(time(1)) + ":" +  fix(time(2)) + ":" +  fix(time(3));
}
```

使用 mouse(3)、mouse(4)获取当前鼠标在坐标系中的坐标。使用 mouse(0)侦测鼠标是否按下。

8.5 图形库

SGC Script 有图形函数，可以选择填入颜色，也可以不填。如果不填颜色，将使用默认颜色代替。有一些内置颜色可以用字符串调用，如红色为"red"。

在运行代码时，图形默认会刷新。使用 refresh()能控制图形是否刷新，如果参数为 true，那么刷新，反之不刷新。如果不传入参数，则会刷新一次。

```
refresh(false);
refresh();
```

使用 clean 清除脚本图层。

```
clean();
```

若要绘制一个点，使用 point：

```
point(1, 2);
```

这将在坐标系中(1,2)处画一个大小为 1px 的点。可以选择颜色：

```
point(1, 2, "red");
```

这将在坐标系中(1,2)处画一个大小为 1px 的、红色的点。

若要绘制线段，使用 line：

```
line(1, 2, 4, 5);
```

这将在坐标系中连接(1,2)、(4,5)。

graph 可以渲染文字。
使用 text 函数显示文字。text(文本,*x*,*y*,size,?,font,color)

```
text("Hello SGC!", 0, 0, 1, 0, 1, 0);
```

使用 rect 画矩形：

```
rect(1, 2, 4, 5);
```

使用 fillrect 填充矩形：

```
fillrect(1, 2, 4, 5, "red");
```

使用 circ 绘制圆：

```
circ(0, 0, 1);
```

使用 fillCirc 填充圆：

```
fillcirc(0, 0, 1, "blue");
```

使用 loadImage 将图像加载到缓存：

```
loadImage("https://sqy419.axolotlpower.com/sgc/v4/logo00.svg", "myImage");
```

使用 Image 显示图像：

```
image("myImage", 0, 0, 100, 0);
```

最后一个参数是透明度，可不填。

8.6 动画播放

SGC Script 有 frame 动画库用于操作 SGC 的逐帧动画。
frameSpeed 设置动画速度，默认值为 1。
frameMax 设置最大帧数，最大为 99999 帧。
frameStart 开始动画，stop 停止动画。

```
frameSpeed(1.14514);
frameMax(6000);
frameStart();
frameStop();
```

8.7 声音播放

SGC Script 有声音函数用于操作 SGC 的音频播放。
loadSound 可以从网络加载音频：

```
loadSound("https://sqy419.axolotlpower.com/download/assets/music/omr.mp3",
"test")                          //从链接加载音频作为 test
```

如果只有一个参数，loadsound 可以从本地加载音频：

```
loadsound("test")           //从本地加载音频命名为 test
```

playSound 函数用于播放音频：

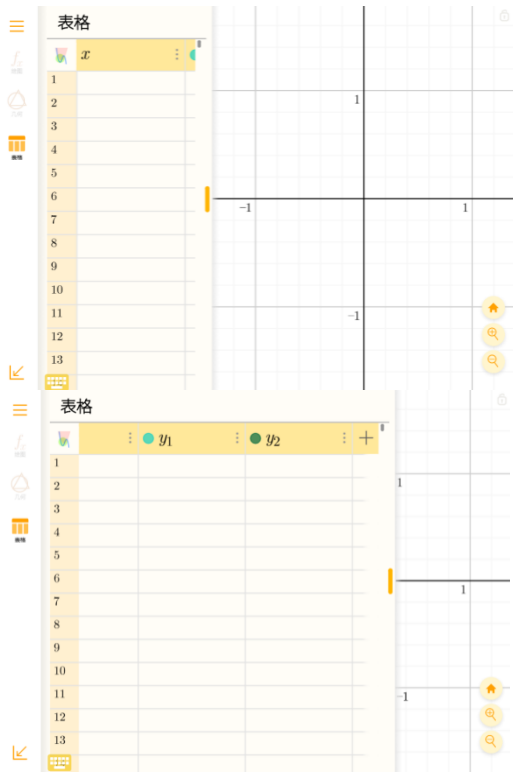
```
playSound("test")
```

9 统计

SGC 支持简易的统计功能，其拥有 16 组双变量统计。

9.1 统计表概述

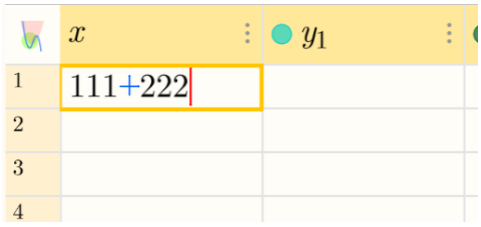
左边模式栏选择表格，出现如图所示的统计页面。您可以使用鼠标拖动统计表。统计表默认有 3 列。



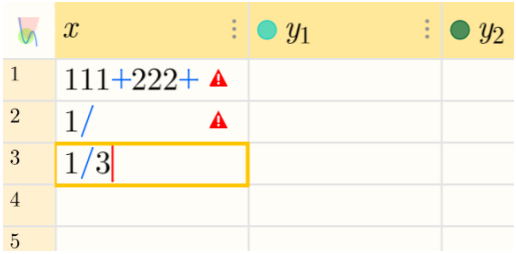
如果在某一列右侧出现了“+”符号，点击它可以新建一列。

9.2 数据的输入

点击一个单元格即可进行输入。被点击的单元格会高亮显示。

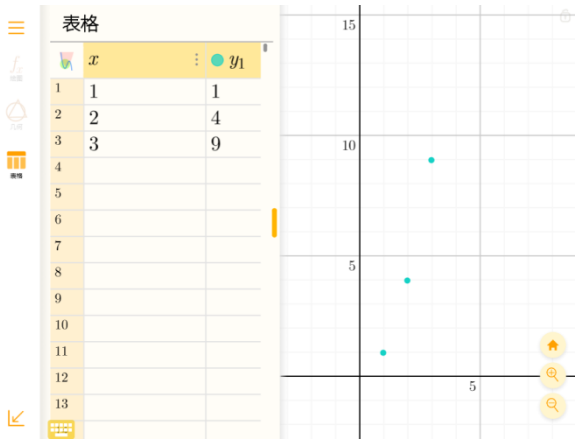


如果输入包含错误，会出现红色警告图标。



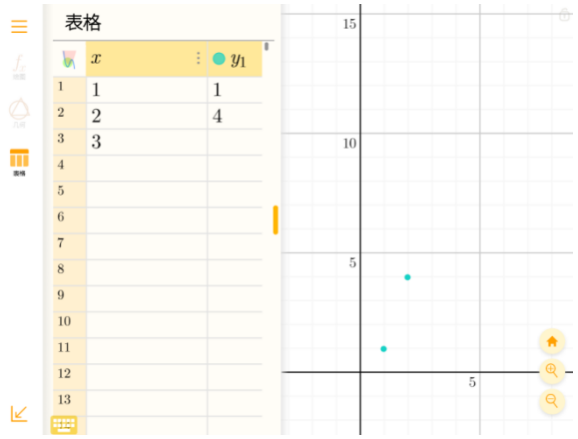
输入方式与输入函数同理，不再赘述。

9.3 数据的显示



(9.3-1)

在输入数据后，坐标系中会出现数据点。这些点的颜色是这一列左侧小圆片的颜色。只有 x、y 均定义的点才会显示。有任意一个不完整的，这个点不会显示。



例如去掉了(3,9)的 9，这个点就没有显示。

9.4 列操作

点击每一列旁边的三个小点，会出现小窗口。

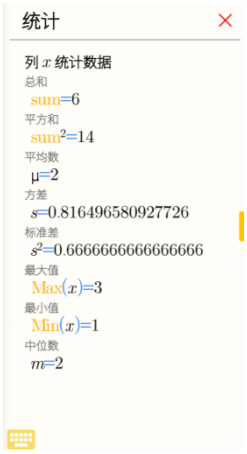


点击“统计”进入列统计模式（**参见 9.5**）。点击“导入”可以导入数据，数据需要以空格间隔。例如某一列为[1,2,3,4,5]，导入输入 1 2 3 4 5 即可。点击“清除列数据”清除这一列的数据。在点击 y_n 列的时候才会显示“回归”功能。点击“回归”启动 y_n 列的回归计算。点击“样式”可以修改数据的显示颜色。

9.5 列统计计算

在点击“统计”后，会对这一列进行统计。出现如图(9.5-1)所示页面。这统计具有如下功能：求 Σx，Σx^2，$\bar{x}$，方差，标准差，Min(最小值)，Max(最大值)，Mid(中位数)。

如果统计数据有误（例如空列表），会给出如图(9.5-2)提示。



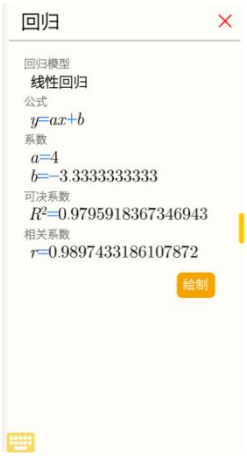
(9.5-1)



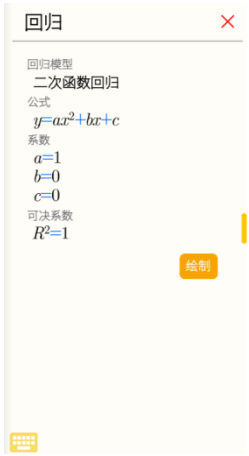
(9.5-2)

9.6 列回归计算

在点击“回归”后，会对这一列和 x 列进行回归计算，出现如图(9.6-1)所示页面。这界面会显示回归的参数，包括回归系数和可决系数。您可以自己选择回归模型。默认是线性回归，支持线性、二次、指数(ae^{bx})、对数回归($a+b\ln x$)。点击“回归模型”一栏就可以进行选择。如对(9.3-1)中的列进行二次回归计算，如图(9.6-2)。

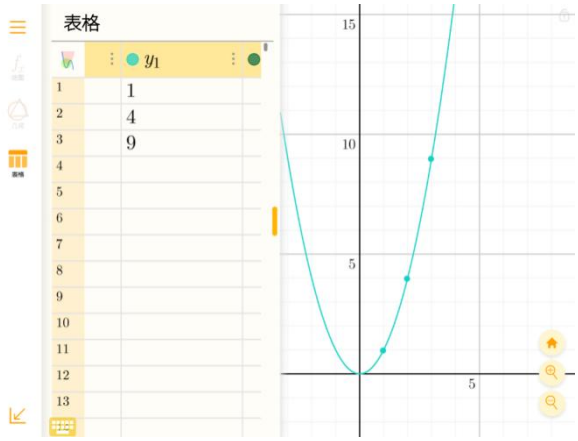


(9.6-1)



(9.6-2)

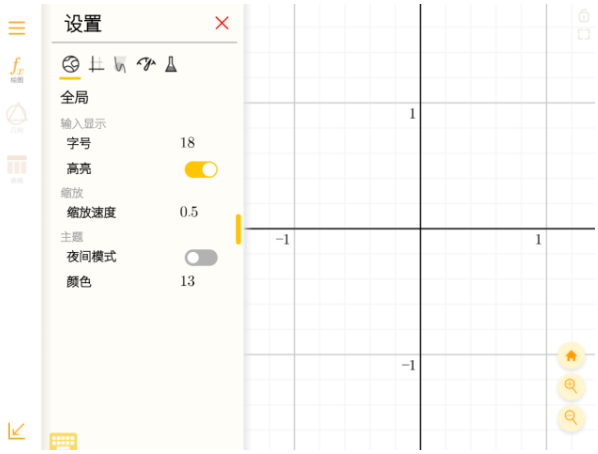
回归计算页面下方有一“绘制”按钮。点击即可将回归结果绘入坐标系。



10 设置

打开系统设置后，设置分为 5 个板块。

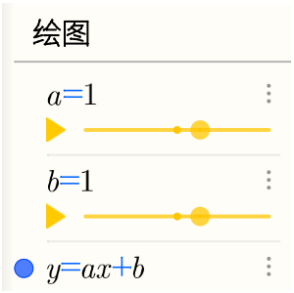
10.1　全局



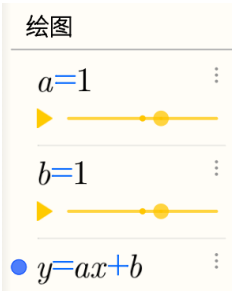
如图。全局设置的符号是一个类似地球的图案。全局里包含一些基本设置。

10.1.1 输入显示

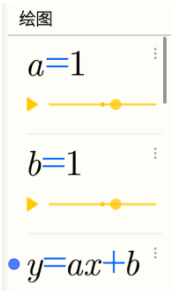
字号：点击字体大小后面的数字可以修改字体大小，最大是 36 点字体。下面是不同字号的对比。



18px



24px

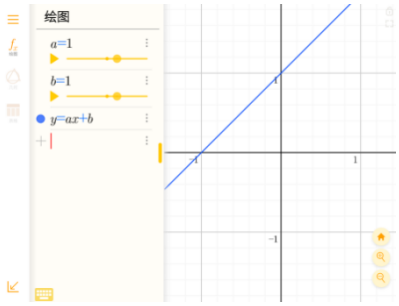


36px

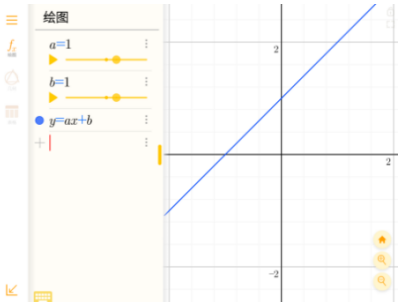
高亮显示前面已经提到过，参见 **2.3**。

10.1.2 缩放

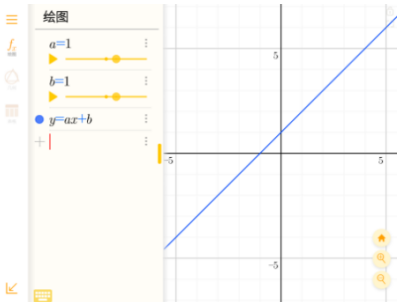
修改缩放速度后，鼠标滚轮的滚动与画布放大缩小按钮的效果将会改变。下面是只点击一次缩放按钮，使用不同缩放速度的效果（坐标系均回到初始位置）：



0.01

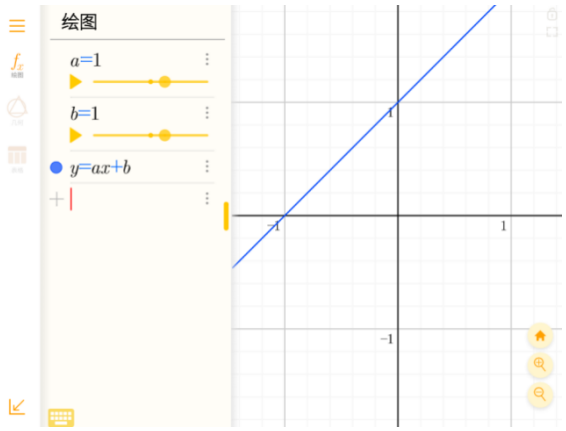


0.5

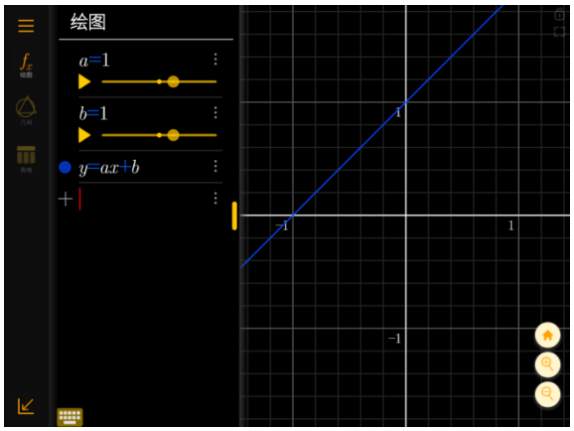


3

主题下可以打开夜间模式。打开后，软件以黑色为主题，但是绘图区的输入颜色不会取反。输入区黑色的字会变成白色。



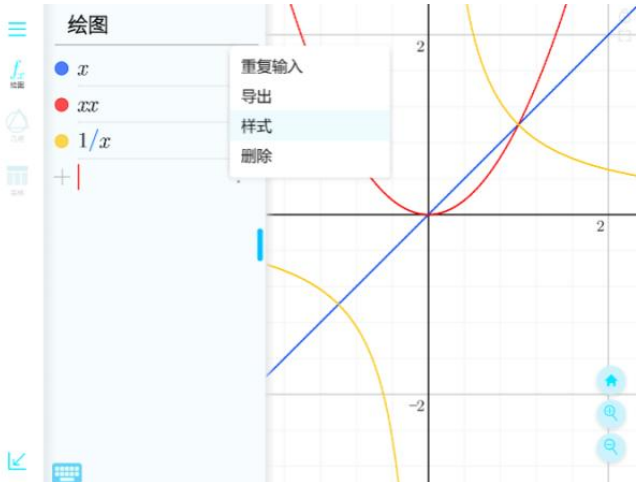
夜间模式：关闭



夜间模式：开启

修改坐标系参数后，右边的图形也会刷新。网格样式点击右侧文本即可修改。

您还可以在此修改颜色主题。SGC 默认颜色主题是黄色，对应颜色值为 13。修改成 54 后效果如图。

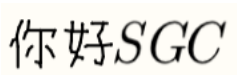


10.1.3 字体

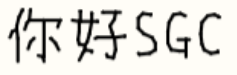
4.10 以上版本可以修改字体。



点击字体名即可更换字体。如果“矢量西文”打开，那么字母、数字等西文仍然使用 CMU Serif 字体渲染，否则使用这个字体的西文部分渲染。例如用小赖字体渲染“你好 SGC”，打开矢量西文：



关闭后：



自定义字体对于输入区和坐标系中所有自定义内容均可用。

10.2 坐标系与网格



如图。这个设置可以调整一些有关坐标系与网格的设置。

这里只给出“网格样式”的示例：



主要



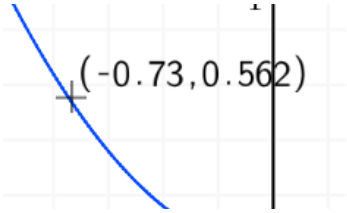
主和次



网格点

如果不显示坐标轴，那么轴的编号会靠侧面显示。

如果打开“显示鼠标坐标”,那么鼠标位置会出现一个十字,这可以指示鼠标的位置,但是不具有吸附功能。



10.3 绘图



如图。绘图里包含一些关于图形的设置。

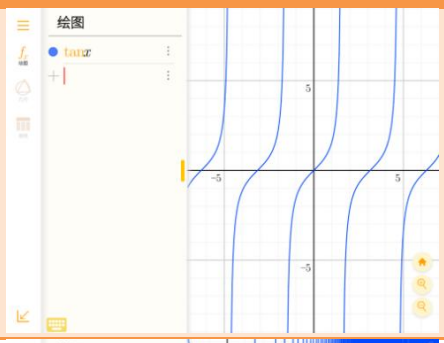
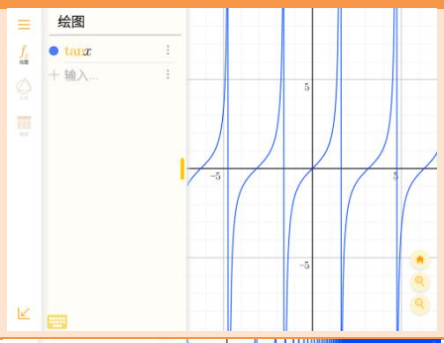
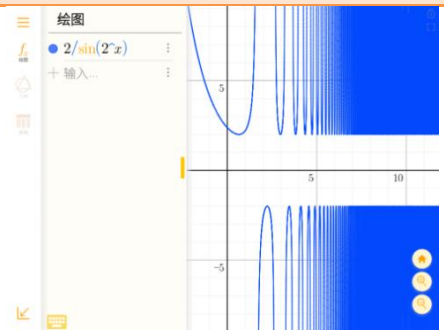
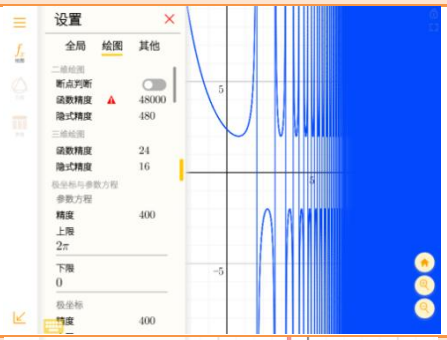
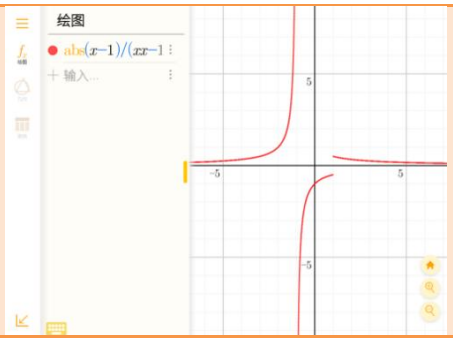
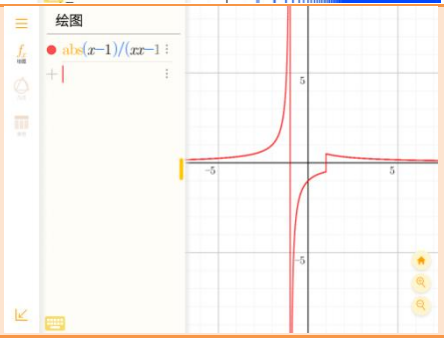
点击断点判断的开关可以关闭或者打开断点判断。由于 SGC 采用描点法作图，因此断点判断对图像质量有很大影响。目前 SGC 能判断一般的显函数的跳跃间断点、无穷间断点和可去间断点。隐函数、参数方程、极坐标函数同样支持断点判断。

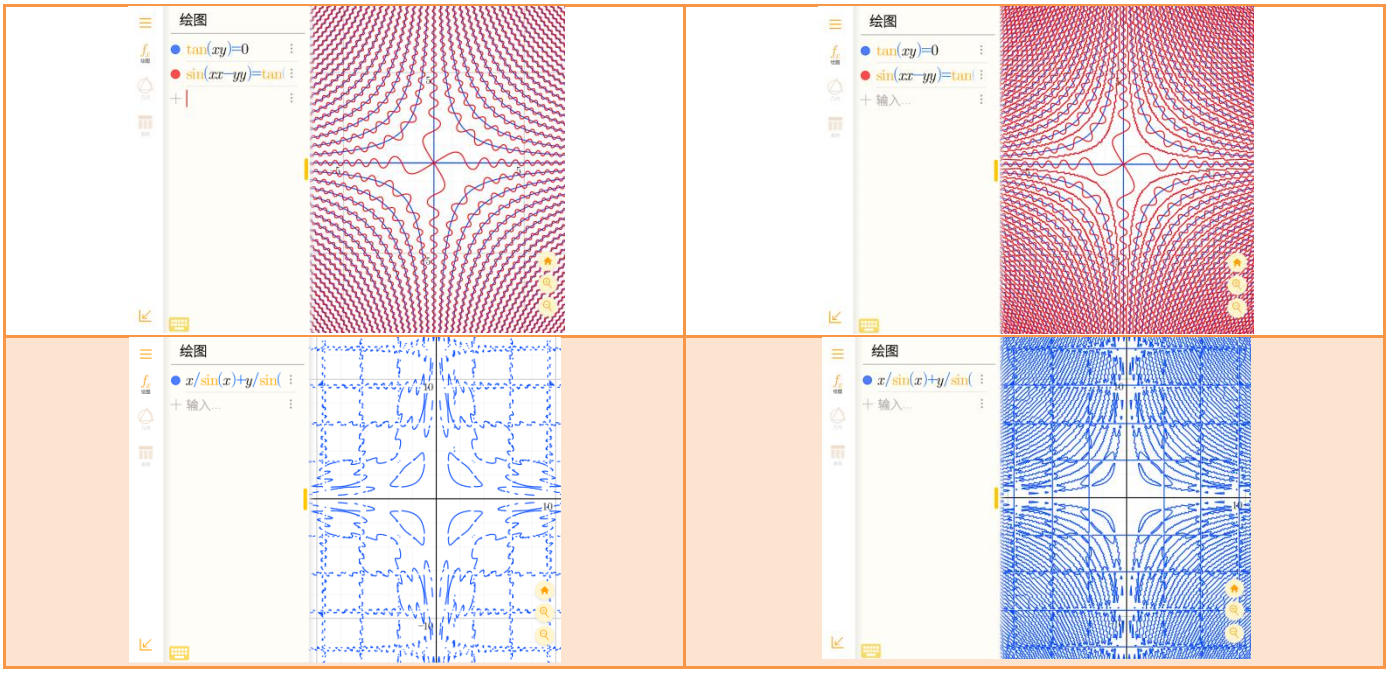
函数精度默认为 480，这表示在 480 个单位中以步长 480/480=1 采集数据点，并连成平滑的线显示在坐标系中。点击后面的数字即可修改函数精度，最大 900000。精度越高，描的点越多，计算就越慢，但图像更精确。

隐式精度为 80，实际是计算横 80 单位、竖 80*0.75=60 个单位、共 80*60=4800 个点。由于使用区块加载，区块大小为 120*120，所以隐式精度必须是 4 的倍数。推荐的精度为 240 至 360，使用 1440 为精度基本上所有图像画的都比 desmos 清楚，但这可能会造成计算缓慢。每次改变精度后，会先初始化隐式缓存单元。

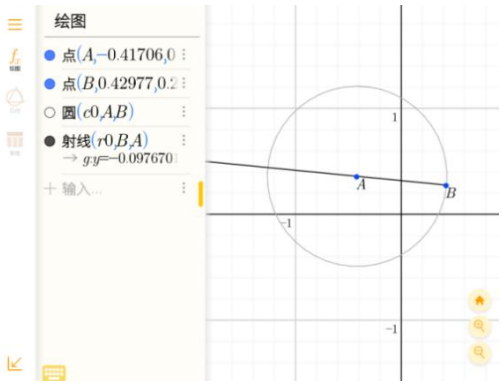
参数方程和极坐标的设置方法同上。（**输入框使用参见 4.4**）

下表列出了断点判断对图像质量的影响。

有断点判断	无断点判断
	
	
	



几何设置中，几何-显示隐藏对象打开后，原本隐藏的看不见的对象会以灰色展示，但是不能被点击到。您可以点击输入区左侧圆形再次显示该对象。如果打开吸附到网格，那么几何中的点如果距离附近某个网格小于 4，将会吸附到最近的网格，同样可以吸附到格点。



如图，圆 *c*0 被隐藏，以灰色显示。

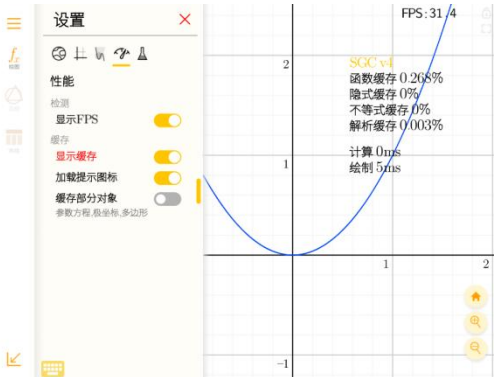
10.4 性能

性能设置里包含一些有关性能的设置。

FPS 显示打开后，FPS 数值会显示在右上角。

显示缓存能显示当前的缓存数据，包括“函数缓存”、“隐式缓存”等，还会显示计算与绘制的时长。

提示图标是一个小沙漏，显示在右上角，如果出现了这个图标，表明正在计算。刷新全局存档时，最上方会出现黄色进度条。



缓存部分对象打开后，除非进行了强制刷新全局存档，这些对象将不会在计算时被计算。这样可以提升图表计算速度，但可能导致图像错误。在参数方程包含正在实时调整的变量时，请不要开启此选项。

10.5 实验功能与其他

如图。其他里包含其他设置和实验性设置。

点击显示特殊点可以在平面模式下，点击函数的极值点或者零点，就能显示其具体位置，建议在函数精度>4800 时使用。

每次点击输入区后，显示的特殊点都会被清除。



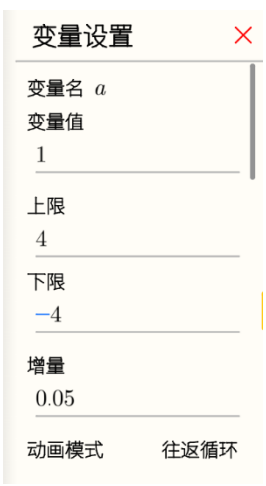
打开输入编号后，输入栏每一栏都会在左侧显示输入编号。



修改变量动画速度后，将会应用到所有变量。

10.6 变量设置

点击输入区想修改的变量右边的三个点，选择设置。





之后进入了变量设置页面。

变量设置页面包含变量名、值等内容。灰色横线以及上面的文字称为文本框单元，如果下面的横线变成了黄色，说明当前正在修改这里的内容，文本框操作方法和输入函数一样。

点击变量名可以修改变量名。

10.6.1 变量设置的修改

点击变量值下文本框单元即可修改变量值，点击 enter 或者空白处和其他输入框，当前输入会自动保存并计算。点击上限或者下限下文本框单元即可修改范围。

注意：如果变量值超出了变量范围，那么会将超出的那个范围替换为变量值的表达式。如果范围超出变量值，会将变量值替换为正在修改的范围表达式。如变量值为 1，却把范围修改成了-1，这会导致变量值变成-1。如果上限比下限小，或下限比上限大，那么上下限会交换位置。

10.6.2 变量增量

修改变量增量后，变量会从当前值开始，依次增加增量，直到大于上限。

动画模式有“往返循环”：变量值从最小增加到最大，又从最大增加到最小，这样重复直到变量停止播放；“单向循环”：变量值从最小增加到最大，又返回最小，再增加到最大，这样重复直到变量停止播放；“单增一次”：从当前值增加到最大，并停止播放。新建的变量默认是往返循环。

常值变量不支持动画。

11 多边形、逐帧动画与批注

11.1 多边形

在 SGC 中，可以创建多边形对象。输入 polygon((x_1,y_1),...,(x_n,y_n))以创建多边形。如果$(x_1,y_1)=(x_n,y_n)$，视为轮廓闭合，那么填充颜色。点的坐标可以包含表达式。

polygon_p 函数只支持点输入，点的坐标不可以包含表达式。polygon_p 函数可以大大加快解析速度。

注意：polygon_g 是几何对象，与这里的 polygon 不同。

11.2 动画

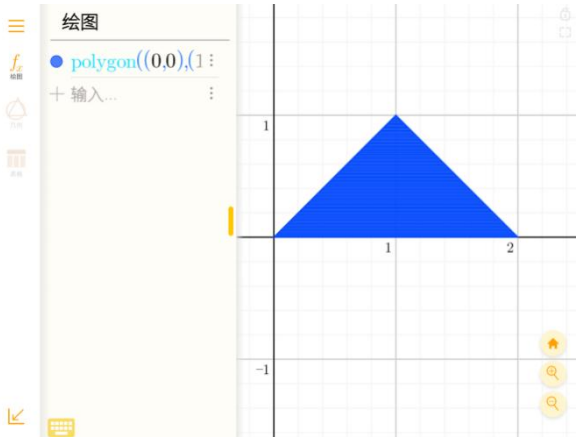
SGC 支持按照帧播放动画。（**打开动画控制台参照章节 1.2**）

11.2.1 创建动画

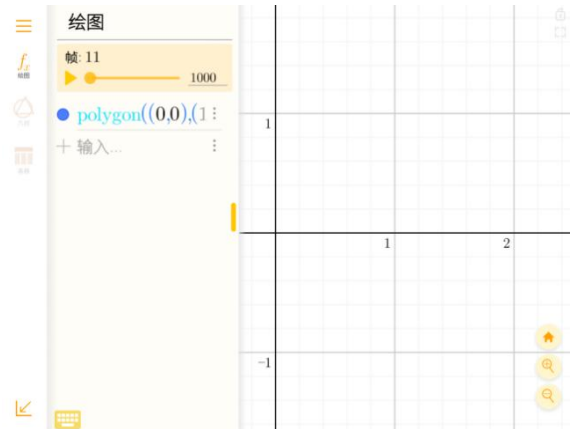
目前，只有 polygon、polygon_p 支持动画。在表达式后面加上#n，就可以在第 n 帧显示这个对象。

如果多边形最后一个点和起始点相同，那么填充这个多边形。输入 polygon((0,0),(1,1),(2,0)) 可以得到一个空心三角形。

输入 polygon((0,0),(1,1),(2,0),(0,0))后：



这会创建一个实心三角形，并且一直显示。修改为 polygon((0,0),(1,1),(2,0),(0,0))#1 后，三角形不显示了。这时，打开控制台，点击播放按钮，发现三角形一闪而过。



11.2.2 开始动画

点击播放按钮开始动画，再次点击可以暂停。

11.2.3 设置

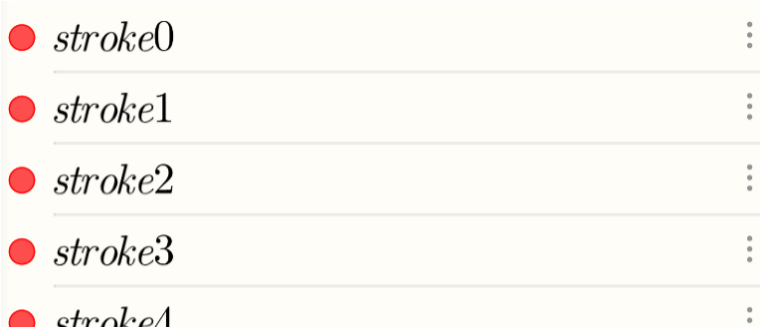
点击“帧：***”可以设置当前帧。点击滑动条右侧的数值可以设置最大帧数。最大为 99999。在设置-绘图-动画中，可以修改动画速度，默认为 1。

11.3 批注

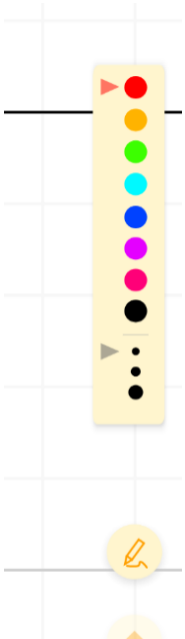
4.8 以上版本支持批注功能。点击右下角“ ”图标即可打开批注。



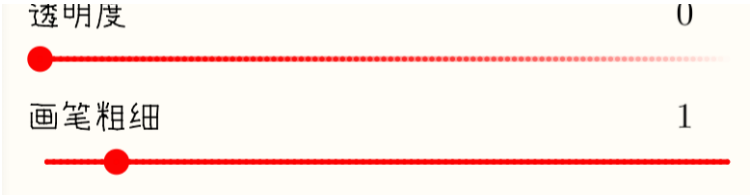
批注会在输入区添加一些以“stroke”开头的内容。



鼠标移动到坐标系右侧，可以展开调色板。这可以修改批注颜色、笔的粗细。



批注颜色、笔的粗细也可以在样式设置面板中修改。



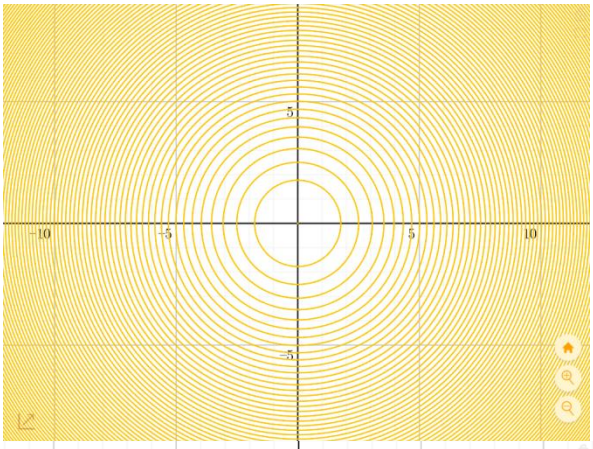
12 附录

12.1 示例表达式

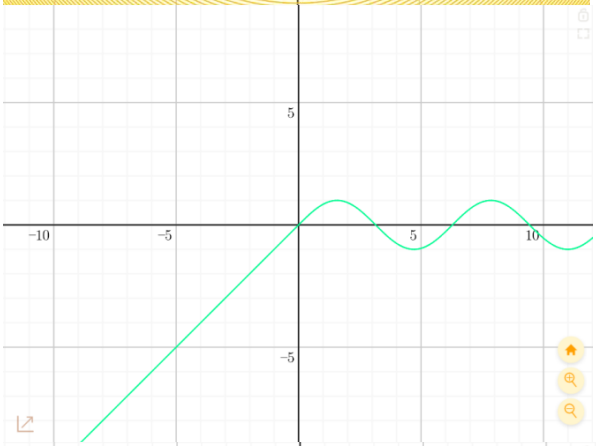
您可以复制以下表达式，使用 SGC 虚拟键盘粘贴到输入区。

二维：

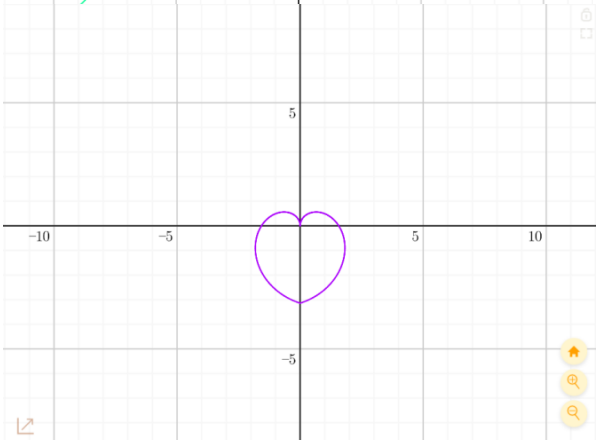
$\sin(xx+yy)=0$



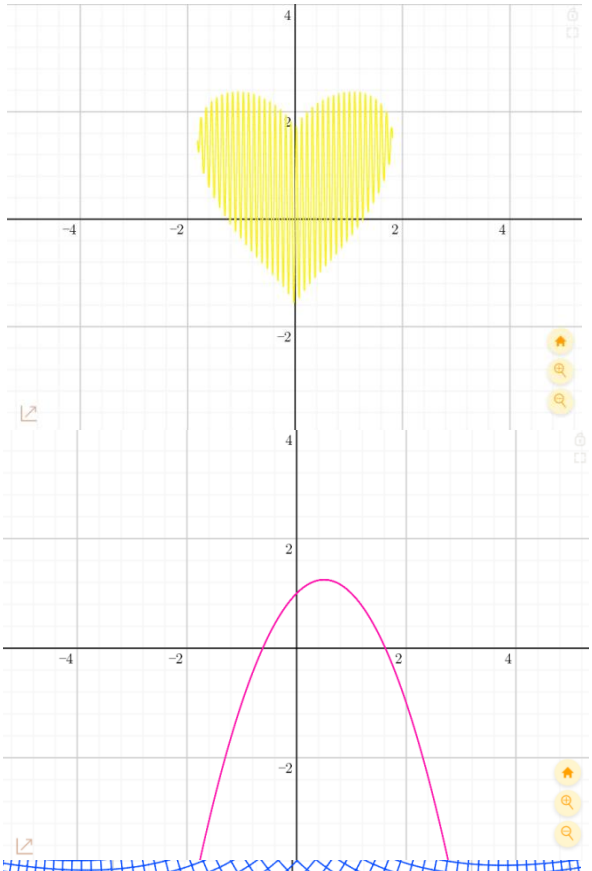
$(x>0)(\sin(x))+x(x<0)$



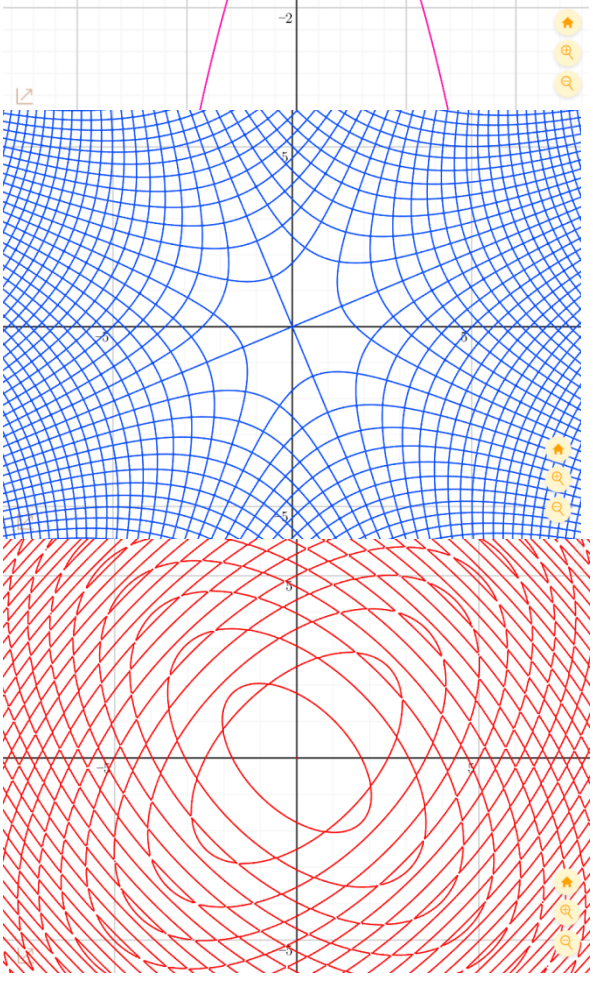
$r=\text{acos}(\sin(\theta))$



a=21.2;
x^(2/3)+0.9sqrt(3.3-xx)sin(3ax)



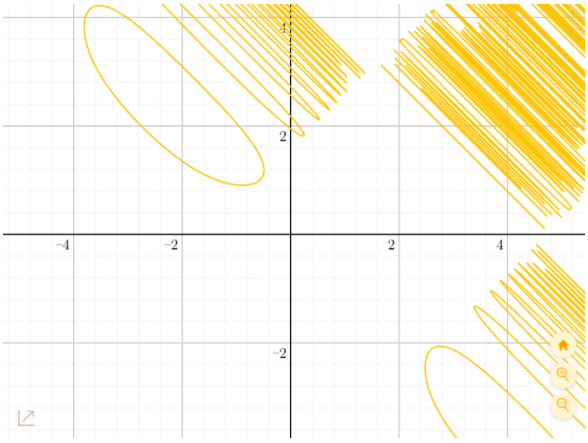
a=-1;
b=1;
c=1;
y=ax^2+bx+c



sin(xx-yy)=sin(2xy)

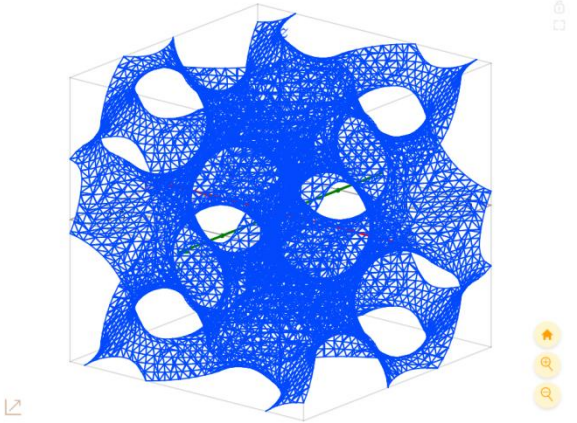
sin(xx+yy)=sin(xy)

$\exp(\sin x+\cos y)=\sin(\exp(x+y))$

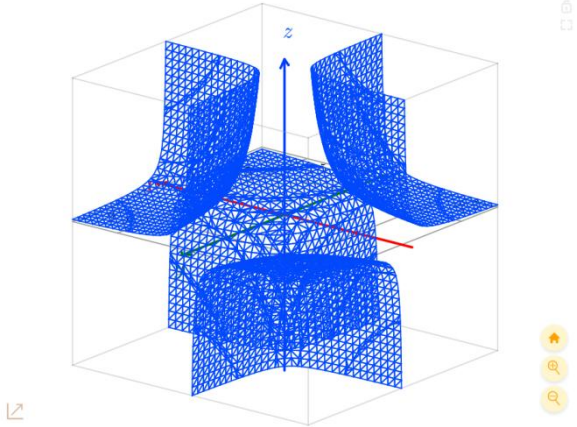


三维:

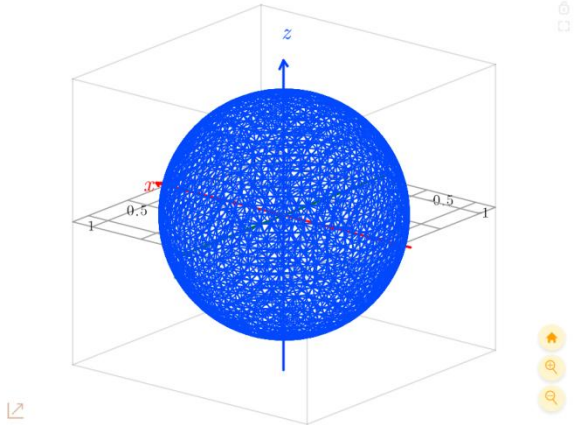
$\exp(\sin x+\cos y)=\sin(\exp(x+y))$



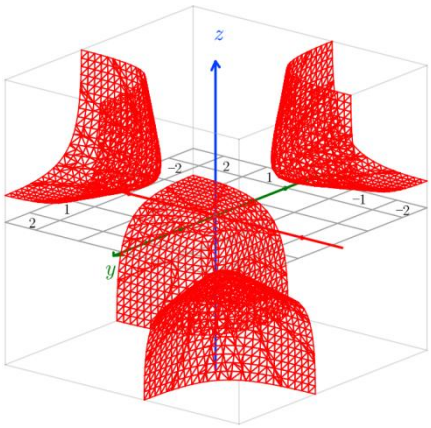
$xyz=1$



$xx+yy+zz=1$

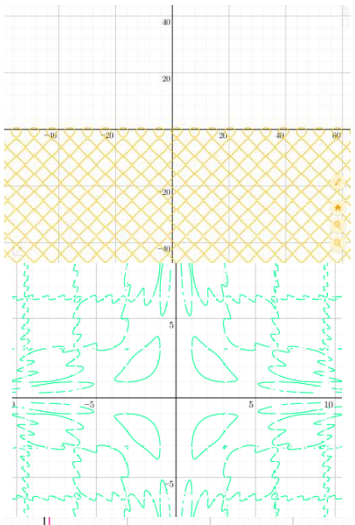
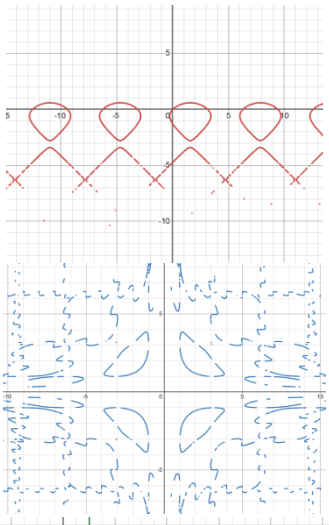
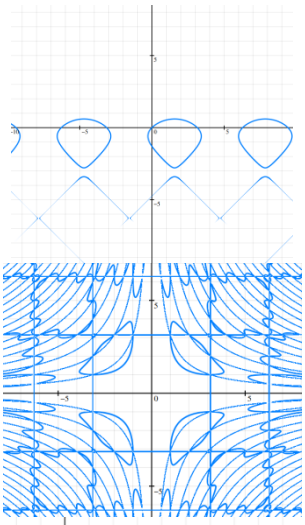
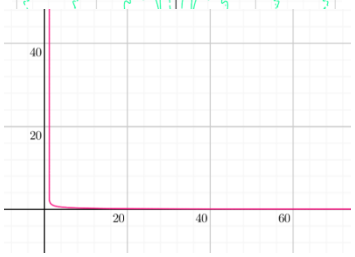
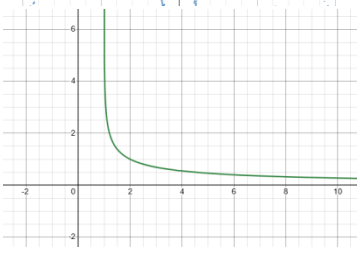
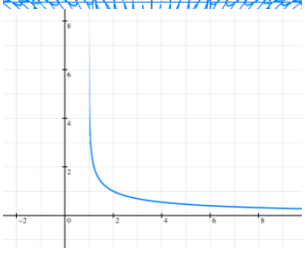


$xx+yy+zz-4xyz=0$



12.2 几个软件对比

这里把几个软件的隐函数绘制做对比。

函数	SGC	Desmos	thetamath
$y=\ln(\sin x+\cos y)$			
$x\csc x+y\csc y=xy\csc(xy)$			
$y=\dfrac{\ln(x-1)}{x-2}$			

13 关于内部实现的一些说明

SGC 是数值计算软件。在此，对一些数值计算方法做一定解释。

13.1 Scratch 的数据类型

计算机中，一般使用单精度(float)或者双精度浮点数(double)表示一个小数。这里，我们介绍双精度浮点数相关的算法，并且 Scratch 使用双精度浮点数。在双精度浮点数中，形如 xe+y 的数等价于 $x\cdot10^y$，如 xe-y 的数等价于 $x\cdot10^{-y}$。双精度浮点数能处理的最小的数是 1e-320 左右。

浮点数计算具有一些问题。想象用双精度浮点数计算 2^{100} +1-2^{100}。我们很容易得出答案是 1。计算机先计算 2^{100}，得到 1.2676506002282294e+30，容易发现，2^{100} 的个位、十位等，因为科学计数法而丢失了。这会导致 2^{100}+1=(1.2676...294e+30)+1，还是等于 1.2676...294e+30。再做减法，导致结果为 0，这不是我们期待的。

13.2 提升三角函数的计算精度

Scratch 特别的一点是，它的 sin, cos, tan 计算精度只有 10 位，但是 asin, acos, atan 的精度却有 15 位。

13.2.1 使用面向+移动

前置知识：在 Scratch 中，面向某个方向是一个独特的东西，方向是角色的右侧朝内的方向。 顺时针为正方向，向上为 0°，向右为 90°，向下为 180°，向左为-90°。由于方向是角色右边朝向的方向，因此 “90°”才是让造型转向和设计页一样的方向。

由单位圆的定义，假设单位圆上一点 A，OA 与 x 轴的夹角为 u，满足 x_A=cos$u$，y_A=sin$u$。于是，可以先让一个角色移动到 (0,0) 处，然后面向 90°–u 方向，并且移动 1 步。那么，这个角色现在的 x、y 坐标就分别是 cos$u$、sin$u$。

13.2.2 使用牛顿迭代法

还可以结合牛顿迭代法来提升 sin, cos, tan 的精度。

sin$x$ 是最小正周期为 2π的函数。于是对于不属于[-π, π]的 x，需要先把 x 取 x 模 2π的值。注意到 sin$x$ 实际上是 arcsin$y$=x 的 “解”。记函数 $f(t)$=arcsin$t$-x，这样，f 的零点便是 sin$x$ 的值。选取 y_0=0，$y_{n+1}=y_n-\frac{f(y_n)}{f'(y_n)}$，多次迭代，那么便可以获取 sin$x$ 的值。简单计算可得$y_{n+1}=y_n-\sqrt{1-y_n^2}(\arcsin y_n-x)$。

由于 Scratch 的 sin, cos, tan 计算精度有 10 位，我们可以选取 Scratch 的 sin$x$ 作为初始值。经测试，最多只需要迭代 1 次就可以得到 15 位精度。例如计算 sin1，实际值为 0.8414709848078965，Scratch 的 sin1 得到 0.8414709848，使用上面的迭代式得到

sin1*=0.8414709848078965。

cos$x$，tan$x$ 的计算同理可得。特别地，cos$x$=1-2sin$^2\left(\frac{x}{2}\right)$。

13.3 数值一阶导数

这里假设以下的函数均具有二阶导数。

在高中，我们学过有关导数的知识。根据导数定义，可导的函数 $f(x)$在 $x=x_0$ 处导数的值，是 $f(x)$在 $x=x_0$ 处切线的斜率。其计算公式是：

$$f'(x)=\lim_{\Delta x\to 0}\frac{f(x+\Delta x)-f(x)}{\Delta x}$$

我们称这样的数值计算方法为向前差分。同样地，也有向后差分：

$$f'(x)=\lim_{\Delta x\to 0}\frac{f(x)-f(x-\Delta x)}{\Delta x}$$

这两种方法都可以用于计算数值导数。这里介绍一种相对来说精度更高的方法：中心差分。

$$f'(x)=\lim_{\Delta x\to 0}\frac{f(x+\Delta x)-f(x-\Delta x)}{2\Delta x}$$

结合导数定义与图像，这公式很好理解。在此我们不做证明。还有很多更高级、精度也更高的公式，由于表述复杂，在此不做解释。

它们的精度如下：

向前差分	$O(h^2)$
向后差分	$O(h^2)$
中心差分	$O(h^2)$+

对于任意的可导的函数 $f(x)$，我们可以使用导数加和、复合函数求导法则等来求导。但是，这对于计算机来说并不好实现（有一种叫做“自动微分”的算法可以实现符号求导，其基本原理是将一般表达式的一个算子 u 变为了<u,u'>来存储，这样使用导数加和、复合函数求导法则等方法时，可以直接获取 u'）。因此，我们可以使用数值求导。使用中心差分就能快速获取导数值，而选取 Δx 的值尤为重要。

并不是越小的 Δx 就越好。如果 Δx 取得过小，会导致$f(x+\Delta x)$、$f(x-\Delta x)$在双精度浮点数中的数值近乎相等，甚至完全一样，数值求导方法就失效了。考虑到 $O(h^2)$的精度，我们可以选取 1e-6 至 1e-5 的 Δx。

例如：想要计算 sin$x$ 在 x=1 处的导数。我们容易求得，这数应该为 cos1，约为 0.5403023058681398。

编写 JavaScript 代码如下：

```
dx=5e-4;
a = ([Math.sin(1+dx),Math.sin(1-dx),
      (Math.sin(1+dx)-Math.sin(1-dx))/dx/2]);
console.log(a);
```

我们选取一些 Δx，用中心差分公式计算，得到如下表：

$f(x+\Delta x)$	$f(x-\Delta x)$	计算结果	Δx
0.8417410307657034	0.8412007284823478	0.54030228335555537	5e-4=0.0005
0.8415250108310384	0.8414169503700448	0.54030230496771033	1e-4=0.0001
0.8414979988713400	0.8414439686407756	0.54030230564383610	5e-5=0.00005
0.8414763877888816	0.8414655817427644	0.54030230585699890	1e-5=0.00001
0.8414736863089075	0.8414682832858488	0.54030230587365220	5e-6=0.000005
0.8414723355610315	0.8414696340495023	0.54030230585144780	2.5e-6=0.0000025
0.8414715251097816	0.8414704445051698	0.54030230589585670	1e-6=0.000001
0.8414712549589443	0.8414707146566384	0.54030230589585670	5e-7=0.0000005
0.8414710388381229	0.8414709307776618	0.54030230567381210	1e-7=0.0000001

其中蓝色为正确部分。不难发现，随着 Δx 的减小，精度逐渐提高；Δx 小于 1e-6 后，精度反而出现了倒退的现象。

综上，我们可以选取 2.5e-6 左右的数作为 Δx。

顺便提一下计算二阶数值导数的方法。注意到$f''(x)$是给 $f(x)$求导后再求导，因此我们可以计算

$$f''(x) = \lim_{\Delta x \to 0} \frac{f'(x+\Delta x) - f'(x-\Delta x)}{2\Delta x}$$

但这可能会导致非常大的误差，因为一阶导数本身具有误差，这样反而把误差放大了。我们可以考虑函数 $f(x)$的泰勒展开：

$$f(x+\Delta x) = f(x) + f'(x)(\Delta x) + \frac{f''(x)(\Delta x)^2}{2}$$

$$f(x-\Delta x) = f(x) - f'(x)(\Delta x) + \frac{f''(x)(\Delta x)^2}{2}$$

将上面两个式子相加，消去$f'(x)$，便得到

$$f(x+\Delta x) + f(x-\Delta x) = 2f(x) + f''(x)(\Delta x)^2$$

移项并化简，得到

$$f''(x) = \lim_{\Delta x \to 0} \frac{f(x+\Delta x) - 2f(x) + f(x-\Delta x)}{\Delta x^2}$$

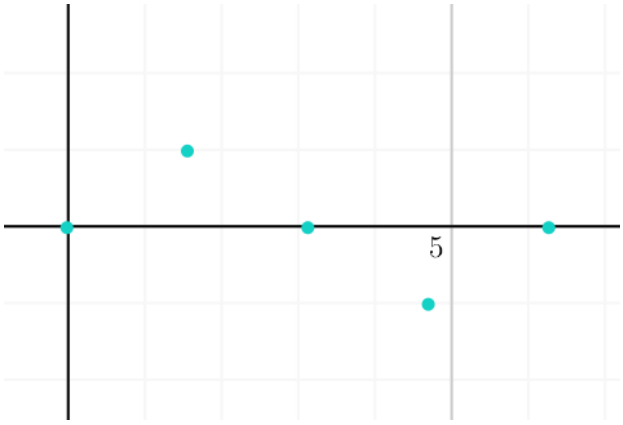
这样的二阶导数不依赖于$f'(x)$的计算，具有较高的精度和简单的计算过程。

13.4 作函数的图像

在初中，我们学过有关函数的知识。作函数的图像一般使用描点法。我们想要画出函数 $f(x)$的图像，可以计算 $f(-2)$、$f(-1.5)$、$f(-1)$，等等。将这些函数的值和对应的 x 放在平面直角坐标系中，用平滑的曲线连接这些点，便得到这函数大致的图像。

在计算机中，一般地，也可以使用描点法作图。鉴于计算机计算的速度比手工计算快得多，我们可以取更多的 x 来计算，这样画出的图像也会更精确。例如，想要画

出 $y=\sin x$ 在区间$[0, 2\pi]$的图像，我们可以取$(0,0)$、$\left(\frac{\pi}{2},1\right)$、$(\pi,0)$、$\left(\frac{3\pi}{2},1\right)$、$(2\pi,0)$这五个点，基于我们学过的 $\sin x$ 的性质，大致就能描绘 $y=\sin x$ 的图像。



但是，我们只使用这五个点作图总会有误差。因此，我们可以编写程序，取更多的 x 值。例如这里选取$[0, 2\pi]$中 101 个 x，分别是 0、$\pi/50$、$\pi/25$、...、2π。使用计算机算得如下结果（这里只展示一部分）：

x	0.8168	0.8482	0.8796	0.9111	0.9425	0.9738
$\sin x$	0.7289	0.7501	0.7705	0.7902	0.8090	0.8271

这样，我们便可以得到更精细的曲线。

以上便是描点法的基本思想。

对于一些变化率比较大的函数，这个方法就不能很好地描绘函数图像。如 $y=\sin(e^x)$，随着 x 的增大，这函数震荡的愈发剧烈。一般地，可以通过增加描点的数量即提高精度来改善，但是效果仍然不理想。考虑到函数的导数能反映这函数的变化率，我们便可以根据导数来调整步长，这称为自适应绘图。

自适应绘图大致的思路如下：我们默认步长为 j。在计算 $f(x_i)$ $(i>2)$时，总能根据导数公式，计算$f'(x_i)$的值。之后，我们便记$|f'(x_i)|=k$，利用 k 的值来调整步长。比如可以让下一次计算的步长为 $1/k$。这样，对于变化率比较大的函数，步长能自动调整，便能较好地描绘图像。

特别地，我们还需要限制 $1/k$ 的最大值与最小值，防止精度过低或者过高，导致计算过多数据。

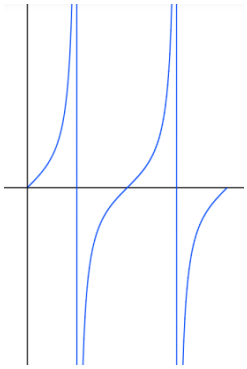
这方法可以进一步优化。每当计算$f'(x_i)$时，总需要至少多计算一次，带来额外的开销。因此，我们可以利用 $f(x_i)$与$f(x_{i-1})$来近似计算$f'(x_i)$。这样能提升计算速度。

SGC v5 便支持使用此方法绘图。但是，这个方法依然会带来一些问题。例如 $y=e^x$ 不需要使用自适应绘图，因为其单调递增，但是其导数仍为 e^x，随着 x 增大，e^x 也增大，导致步长变小，计算了无意义的数据。

13.5 数值断点判断

上述的基本描点法看似没有问题，但计算机并不能很容易地获取函数的性质。例

如：$y=\tan x$ 在区间$[0, 2\pi]$上具有无穷间断点。我们可以很容易知道在 $x=\frac{\pi}{2}$和 $x=\frac{3\pi}{2}$处有间断点，可以在此处不连接。但是使用基本的描点法，我们得到如下“错误”图像：



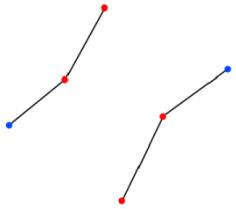
计算机并不知道在 $x=\frac{\pi}{2}$和 $x=\frac{3\pi}{2}$处有间断点，因此会在此处连接。这种情况应该如何处理呢?

根据间断点的部分定义，我们知道：如果函数 $f(x)$在 $x=x_0$ 处具有不相等的左极限和右极限，这 x_0 便是函数 $f(x)$的一个间断点。因此，我们可以在每一次计算第 i 个 x 时，考虑 $f(x_{i-1})$和 $f(x_{i+1})$。如果 $f(x_i)$未定义，那么 x_0 就是间断点。如果这两个数相差超过了 ε，那么便认为这是一个间断点。ε 可以取 1e-2 左右，根据步长 i 而定。

第二种方法是计算 $f(x_i+\Delta x)$与 $f(x_i-\Delta x)$来替代 $f(x_{i-1})$和 $f(x_{i+1})$。第三种方法是取 $|f(x_{i-1})-f(x_i)|$与 ε 比较。SGC v1、v2 采用此方法。

上述三种方法并不具有普遍性，并且很容易误判。例如函数 $f(x)=114514x$，就算取了较小的步长，$f(x_{i-1})$和 $f(x_{i+1})$的差始终都会很大，导致所有点全部断开，这不是我们所期望的。这里介绍一种新的方法：四点法，也是 SGC v4 所采用的方法。

(1)考虑无穷间断点

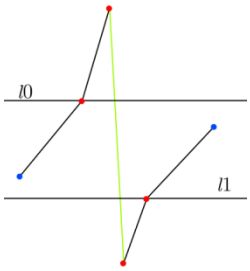


如图所示，如 $y=\tan x$ 在 $x=\frac{\pi}{2}$附近的行为。

取标红色的四个点，从左到右按顺序记为 v_1、v_2、v_3、v_4。

如果 v_1、v_2、v_3、v_4 中有未定义值，认为是间断点。

如图，v_1-v_2 与 v_3-v_4 的符号相同，v_2-v_3 与 v_1-v_2、v_3-v_4 中任意一个符号相反，即 $(v_2-v_3)(v_1-v_2)\leqslant 0$ || $(v_2-v_3)(v_3-v_4)\leqslant 0$

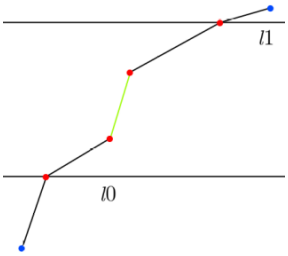


过 v_1、v_4 做 x 轴平行线 $l0$、$l1$。如果 $l0$ 在 $l1$ 的上方，认为 v_2-v_3 中有间断点。表达式为 $|v_1-v_2|+|v_3-v_4|<|v_2-v_3|$。保险起见，我们加上误差 ε，即需要满足 $|v_1-v_2|+|v_3-v_4|+\varepsilon<|v_2-v_3|$。这个式子无论什么断点都须要满足，其中 ε 视步长而定。SGC 中，取的是 ε=0.01/当前缩放。

(2) 考虑跳跃间断点

例如定义分段函数 $f(x)=\begin{cases}\sin x, & x<1\\ x, & x\geq 1\end{cases}$。

显然，此函数在 x=1 处产生跳跃间断点，并且使用(1)中做差却产生了两侧同符号。

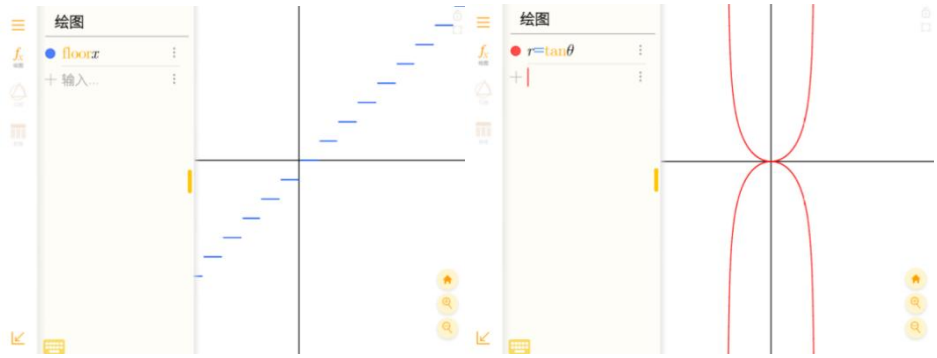


这里我们使用第二做差法：$|(v_1-v_2)(v_3-v_4)|<\varepsilon|v_2-v_3|$。$\varepsilon$ 视步长而定。SGC 中，取的是 ε=0.015/当前缩放。

(3) 考虑震荡间断点

如 $y=\sin\frac{1}{x}$ 在 x=0 附近的行为。一般情况下，震荡间断点不易被绘制出来，在此我们不做讨论。

使用(1)解决了 $y=\tan x$、$y=\frac{1}{x}$ 和 $y=[x]$ 这类函数的间断点问题。使用(2)解决了分段函数的间断点问题。



极坐标、参数方程的间断点判断方法同理可得。

13.6 数值积分

数值积分已经有很多方法计算，例如矩形法、梯形法、抛物线法(Simpson)等。但是这些方法对于较复杂的被积函数，收敛速度就显得慢了。

这里介绍两种收敛较快、精度较高的方法。

(1)龙贝格法(Romberg)

将变步长（区间数加倍、步长减半）梯形法的积分近似值通过线性组合得到变步长 Simpson 法的积分近似值,再将变步长 Simpson 法的积分近似值通过线性组合得到变步长 Cotes 法的积分近似值，再将变步长 Cotes 法的积分近似值通过线性组合得到变步长 Romberg 法的积分近似值。这种逐次二分积分区间并利用线性组合的加速方法将一个粗糙的积分值 T_n序列逐步加工成精度较高的积分值 R_n序列的方法，称为 Romberg 积分法。

由于龙贝格法的步骤较多，且 Scratch 难以实现，在此不做过多介绍。

(2)高斯积分法与 Gauss-Kronrod 法

看梯形法和抛物线法的表达式，会发现这两个方法的结果可以理解为一系列点的函数值，乘上某个系数后求和。这两种方法都是均匀分割区间来进行求和，至少具有 n-1 的代数精度。而高斯积分法并不是均匀分割区间,而是计算不同高斯节点，乘以每个点的权重后再求和。它具有 $2n+1$ 的代数精度，可精确处理 $2n+1$ 次以下多项式积分。

高斯积分法处理[-1,1]上的$f(x)$。我们可以用区间变换实现处理[a,b]。这里不介绍高斯积分法具体的数学原理，只解释高斯积分法如何进行数值计算。高斯积分法可理解为以下公式：

$$\int_{-1}^{1} f(x)\mathrm{d}x=\sum_{j=0}^{n} w_j f(x_j)$$

x_j是节点，w_j是权重。

高斯积分法节点使用勒让德多项式计算。勒让德(Legendre)多项式满足以下性质：当某个多项式 $P(x)$的次数小于 n 时，有

$$\int_{-1}^{1} P(x)P_n(x)\mathrm{d}x=0$$

并且满足递推关系：

$$P_{n+1}(x)=\frac{2n+1}{n+1}xP_n(x)-\frac{n}{n+1}P_{n-1}(x)\ (n=1,2,\ldots)$$

这样，我们便求得

$$P_1(x)=x$$

$$P_2(x)=\frac{3x^2-1}{2}$$

$$P_3(x)=\frac{5x^3-3x}{2}$$

以此类推。n 阶高斯积分法的节点便是$P_n(x)$的根。注意到，$P_n(x)$的根全部处于[-1,1]上，它具有 n 个实根。n 阶高斯积分法的系数有如下公式：

$$w_i=\frac{2}{(1-x_i^2)\left(P_n'(x_i)\right)^2}$$

例如：要计算

$$\int_{-1}^{1}x^2\mathrm{d}x$$

这里我们选用 2 阶高斯积分公式。先求

$$P_2(x)=\frac{3x^2-1}{2}$$

的根，注意到是

$$\left\{-\frac{\sqrt{3}}{3},\frac{\sqrt{3}}{3}\right\}$$

也就是{-0.5773502691896257,0.5773502691896257}。

再求出权重 w={1,1}。代入得

$$\int_{-1}^{1}x^2\mathrm{d}x\approx\left(-\frac{\sqrt{3}}{3}\right)^2+\left(\frac{\sqrt{3}}{3}\right)^2=\frac{2}{3}$$

这方法只计算了 2 次得到了准确结果。再例如使用 5 阶高斯积分公式计算

$$\int_{-1}^{1}\cos x\mathrm{d}x$$

其结果应该为 2sin1=1.682941969615793。这里我们计算节点为

$$\left\{0,\pm\frac{1}{3}\sqrt{5-2\sqrt{\frac{10}{7}}},\pm\frac{1}{3}\sqrt{5+2\sqrt{\frac{10}{7}}}\right\}$$

其数值为{0, ±0.538469310105683, ±0.906179845938664}。

计算权重为

$$\left\{\frac{128}{225},\frac{322+13\sqrt{70}}{900},\frac{322+13\sqrt{70}}{900},\frac{322-13\sqrt{70}}{900},\frac{322-13\sqrt{70}}{900}\right\}$$

其数值为{0.5688888888888889, 0.47862867049936647,
0.47862867049936647, 0.23692688505618908, 0.23692688505618908}。

我们计算

0.5688888888888889·cos0
+2·(0.47862867049936647·cos0.538469310105683
+0.23692688505618908·cos0.906179845938664)
=1.682941970407192。

想要对函数$f(x)$在区间$[a,b]$积分，只需要做变换：

$$\int_a^b f(x)\mathrm{d}x=\frac{b-a}{2}\int_{-1}^1 f\left(\frac{b-a}{2}t+\frac{a+b}{2}\right)\mathrm{d}t$$

以上便是高斯积分法的基本思路。在实际应用时，我们可以提前计算好节点和权重，之后直接调用。SGC v3 就采用 300 节点的高斯积分进行概率计算。

不难发现，仅仅使用固定阶高斯积分法仍然存在问题。例如计算

$$\int_{-1}^1 |\sqrt{x}|\mathrm{d}x$$

便遇到了收敛缓慢的问题。即便取 500 节点，精度也不是很高。这里，我们可以借鉴龙贝格法的思路：

(1)计算某区间的值 G_n

在区间 u 计算 n 阶高斯积分法 G_n 的结果后，我们可以取更高的阶 G_{n+1} 进行第二次计算，然后比较$|G_n\text{-}G_{n+1}|$是否大于 ε。

(2)误差判断

$|G_n\text{-}G_{n+1}|$如果大于 ε，这说明这个区间是 n 阶不能处理的。这时，便可以二分区间，提高精度。在二分的两个区间中，分别使用(1)中的方法再次进行判断。最后将各区间结果累加即可。

二分法一般采用递归的思路。重要的一点是，我们还需要设置递归上限，防止死循环。

注意到，计算 G_{n+1} 又需要计算额外的节点，造成计算量巨大。我们需要一种方法，重复利用前面 G_n 的部分结果来减小计算量。这里介绍 Gauss-Kronrod 方法。Gauss-Kronrod 方法基于 Kronrod 点的误差估计，可充分利用现有计算值，从而达到有效控制精度的同时，性能没有太大的损失。公式可理解为

$$\int_{-1}^1 f(x)\mathrm{d}x=\sum_{j=0}^{n} w_j f(x_j)+\sum_{j=0}^{n+1} v_j f(u_j)$$

u_j是 Stieltjes-Wigert 多项式的根。具体计算方法这里不给出。

数值积分重要的一点是大致判断积分的敛散性。这一点较为简单。在递归过程中，如果连续多次出现$|G_n\text{-}G_{n+1}|>\varepsilon_1$，而且这个 ε_1 较大，那么认为此积分发散。

13.7 一元方程的数值解与 fSolve 函数

数值解有多种方法计算，如高中所学的二分法等。但是二分法收敛可能较慢，这里我们介绍几种其他方法：

(1)牛顿迭代法(Newton-Raphson 迭代法)与割线法

要求解 $f(x)=0$，选取初始值 x_0，可以考虑 $f(x)$在 $x=x_0$ 泰勒展开的线性部分：

$$f(x) = f(x_0) + f'(x_0)(x - \Delta x)$$

因此令

$$0 = f(x_0) + f'(x_0)(x - x_0)$$

作为方程根的近似。解得每一个新的 x 是

$$x_{n+1}=x_n - \frac{f(x_n)}{f'(x_n)}$$

用这公式解 x^2-x-1=0，令 $f=x^2$-x-1，并选取 x_0=1 迭代，直到$|x_n$-$x_{n+1}|<\varepsilon$。那么

$$x_{n+1}=x_n - \frac{x_n^2 - x_n - 1}{2x_n - 1}$$

可得

x_1	2.0000000
x_2	1.6666667
x_3	1.6190476
x_4	1.6180344
x_5	1.6180339
x_6	1.6180339

可见这方法收敛速度快且易于实现。导数可以使用前面介绍过的方法进行计算。SGC v4 的 fSolve 函数采用此方法。

但是，牛顿法的震荡问题仍没有很好的解决方法。

割线法是牛顿法的改进，是利用$f(x_{n-1})$的值和$f(x_n)$进行近似的导数计算。

$$x_{n+1}=x_n - f(x_n)\frac{x_n - x_{n-1}}{f'(x_n) - f(x_{n-1})}$$

(2)复合法

复合法并不需要计算$f'(x_n)$的值。公式如下：

$$x_{n+1}=x_n - \frac{f(x_n)^2}{f(f(x_n)+x_n) - f(x_n)}$$

用这公式解 x^2-x-1=0，并选取 x_0=2，可得

x_1	1.7500000
x_2	1.6388888
x_3	1.6186432
x_4	1.6180345
x_5	1.6180339
x_6	1.6180339

这方法比较依赖于初始值的选择，并不是特别好处理。若选取 x_0=1 进行迭代，则产生了未定义结果。

13.8　特殊函数的数值计算

特殊函数有很多，例如 SGC 中已经实现的 FresnelS、FresnelC、sinIntegral 等。

13.8.1 菲涅尔积分 FresnelS、FresnelC

菲涅耳积分是物理光学、微波技术与天线设计领域的特殊函数。这里我们讲它们的数值计算方法。它们都是奇函数，因此我们只需要考虑 $x>0$ 的计算。

定义

$$\mathrm{FresnelS}(x)=\int_0^x \sin\left(\frac{\pi}{2}t^2\right)\mathrm{d}t$$

$$\mathrm{FresnelC}(x)=\int_0^x \cos\left(\frac{\pi}{2}t^2\right)\mathrm{d}t$$

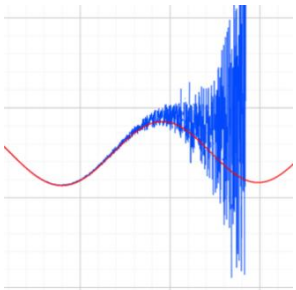
考虑到被积函数在 x 较大时较为密集，这里我们并不使用 Gauss-Kronrod 方法直接进行自适应积分计算，因为这会导致计算缓慢。

注意到它们的级数：

$$\mathrm{FresnelS}(x)=\sqrt{\frac{2}{\pi}}\sum_{n=0}^{\infty}(-1)^n\frac{\left(\frac{\sqrt{2\pi}}{2}x\right)^{4n+3}}{(2n+1)!(4n+3)}$$

$$\mathrm{FresnelC}(x)=\sqrt{\frac{2}{\pi}}\sum_{n=0}^{\infty}(-1)^n\frac{\left(\frac{\sqrt{2\pi}}{2}x\right)^{4n+1}}{(2n)!(4n+1)}$$

这级数对于$x\in\mathbb{R}$均收敛。但是，对于较大的 x，这级数收敛较为缓慢。如图是对于 FresnelC，求和上限为 99 的情景：

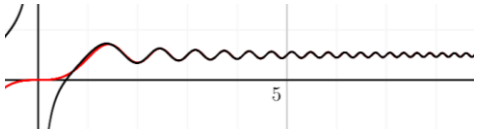


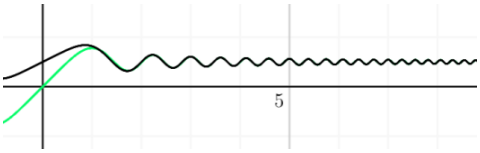
实际计算中，取这么大的上限会导致计算缓慢。考虑到对于$x\gg1$，这两个函数在$x\to\infty$具有渐近展开：

$$\mathrm{FresnelS}(x)\sim\frac{1}{2}-\frac{1}{\pi x}\cos\left(\frac{1}{2}\pi x^2\right)$$

$$\mathrm{FresnelC}(x)\sim\frac{1}{2}+\frac{1}{\pi x}\sin\left(\frac{1}{2}\pi x^2\right)$$

不难绘制出它们的 2 阶展开图像：





注意到，这渐近展开具有较好的拟合。因此，我们可以分段进行计算。对于较小的 x，我们使用普通 Gauss 方法或者级数计算积分；对于较大的 x 则使用渐近展开。20 阶 Gauss 节点即可达到较高的精度。

经测试，x>4.59 即 1.46π 时，即可使用渐近展开。这样的计算原理如下面 JS 代码：

```
xmax = 4.59;
function FresnelS(x){
    if(x < 0){
        _func_temp = -FresnelS(-x);
    } else if (x>xmax){
        //使用渐近展开
        _func_temp= 0.5-Math.cos(0.5*3.141592653589793*x*x)
              /(3.141592653589793*x);
    } else {
        //使用 Gauss 方法积分
    _func_temp_1 = 0;
    _func_temp=0;
    for(let i= 0; i<GaussNode.length; i++) {
    _func_temp_1 = 0.5 * x * (1 + GaussNode[i]);
    _func_temp+= GaussWeight1[i]*
                  0.5 * x *Math.sin(_func_temp_1 *
                  _func_temp_1 * 1.5707963267948966);
            }
        }
    }
    return _func_temp;
}
```

FresnelC 的计算同理可得，这里不给出代码。

事实上，利用 Hornel 格式，可以加速渐近展开计算：

$$\text{FresnelS}(x)\sim\frac{1}{2}+\frac{1}{x^{13}}\left(\left(\left(-\frac{x^4}{\pi}+\frac{3}{\pi^3}\right)x^4-\frac{105}{\pi^5}\right)x^4+\frac{10395}{\pi^7}\right)\cos\left(\frac{\pi}{2}x^2\right)$$
$$+\frac{1}{x^{15}}\left(\left(\left(-\frac{x^4}{\pi^2}+\frac{15}{\pi^4}\right)x^4-\frac{945}{\pi^6}\right)x^4+\frac{135135}{\pi^8}\right)\sin\left(\frac{\pi}{2}x^2\right)$$

这里给出其数值形式

0.5

+(((-0.318309886183791x^4+0.0967546032995985)x^4-0.343115182520605)x^4+3.44171880544581)/x^{13}cos(1.5707963267948966x^2)

+(((-0.101321183642338 x^4+0.153989733820265) x^4-0.982952592264580)x^4+14.2419305760949)/x^{15}sin(1.5707963267948966x^2)

这样便得到精度较高的 FresnelS 渐近展开。对比 WolframAlpha 的结果，得到

x	WolframAlpha	SGC v4	JavaScript
0.5	0.0647324328599	0.0647324328541	0.0647324328599
1	0.4382591473903	0.4382591473851	0.4382591473903
3	0.4963129989673	0.4963129989544	0.4963129989673

13.8.2 LambertW 函数

LambertW 函数可用于求解许多包含指数的超越方程，它是$f=xe^x$的反函数。在 SGC v3 及以前，LambertW 函数简写为 ltW 参与计算。在一些地方（比如 wolfram）也可以使用 ProductLog 代替。

注意到，当 $x>-1$ 时，xe^x 单增；$x<-1$，xe^x 单减；$x=1$ 时取到极值$-e^{-1}$。那么，LambertW 函数应该有 2 个分支。这里，我们实现 LambertW 的主分支，也就是反函数中 $y>-1$ 那部分。LambertW 的主分支定义在$x\in\left[-\frac{1}{e},\infty\right)$上。利用其定义，我们可以使用牛顿迭代法：

$$a_{n+1}=a_n-\frac{ae^a-x}{ae^a+e^a}$$

不妨取 $a_0=1$，这里我们计算 LambertW(1)和 LambertW(100)：

LambertW(1)=0.56714329041；LambertW(100)=3.38563014029

a_1	0.6839397	18.893972
a_2	0.5774544	17.944238
a_3	0.5672297	16.997025
a_4	0.5671432	16.052590
a_5	0.5671432	15.111232
a_6	0.5671432	14.173303

不难发现，LambertW(100)的计算收敛极其缓慢，这可能是由于初始值估计较差。我们选取另一个较好的近似：

$$\text{LambertW}(x)\sim\frac{\sqrt[3]{x}}{2}+0.95\ \ (x\in[20,200])$$

令 $a_0=\frac{\sqrt[3]{100}}{2}+0.95=3.27079441681$ 得

a_1	3.394150903
a_2	3.385674553
a_3	3.385630145
a_4	3.385630140(29)
a_5	3.385630140(29)

这便有很高的精度。但是对于$x\gg 200$，就算初始值比较精确，也不起作用了。我们可以使用对数迭代：

$$a_{n+1}=a_n-\frac{a\ln a-x}{\ln a+1}$$

迭代完成后，返回 $r=\ln(a_n)$。

例如计算 LambertW(10000000)=13.5143440103：

令 $a_0=\frac{\sqrt[3]{10000000}}{2}+0.95=108.671734502$，迭代得

a_1	1758003.77856	$\ln a_1$	14.3796895
a_2	764515.029613	$\ln a_2$	13.5469969
a_3	739981.939680	$\ln a_3$	13.5143811
a_4	739954.524825	$\ln a_4$	13.5143440
a_5	739954.524790	$\ln a_5$	13.5143440(1)

得到正确结果。

13.8.3 Gamma 函数与 Psi 函数

Gamma 函数作为阶乘函数的解析延拓，具有重要意义。

除了负整数，它的定义是：

$$\Gamma(x)=\int_0^{\infty}\mathrm{e}^{-t}t^{x-1}\mathrm{d}t$$

于是 $x!=\mathrm{Gamma}(x+1)$。我们一般不使用原始定义来计算 Gamma 函数。常见的方法有 Lanczos 近似。其公式如下：

$$\Gamma(z)=\sqrt{2\pi}\left(z+g+\frac{1}{2}\right)^{z+\frac{1}{2}}\mathrm{e}^{-z-g-\frac{1}{2}}\sum_{k=0}^{n}\frac{p_n}{z+k}$$

常见的一组 p 如下：($g=7, n=9$)

$$p_0=0.99999999999980993$$
$$p_1=676.5203681218851$$
$$p_2=-1259.1392167224028$$
$$p_3=771.32342877765313$$
$$p_4=-176.61502916214059$$
$$p_5=12.507343278686905$$
$$p_6=-0.13857109526572012$$
$$p_7=9.9843695780195716\text{e-}6$$
$$p_8=1.5056327351493116\text{e-}7$$

$$\Gamma(x)\sim\sqrt{2\pi}\left(x+\frac{13}{2}\right)^{x-\frac{1}{2}}\mathrm{e}^{-\left(x+\frac{13}{2}\right)}\left(p_0+\frac{p_1}{x}+\frac{p_2}{x+1}+\frac{p_3}{x+2}+\frac{p_4}{x+3}+\frac{p_5}{x+4}+\frac{p_6}{x+5}+\frac{p_7}{x+6}+\frac{p_8}{x+7}\right)$$

此公式随着 x 增大，精度降低。

x	100	1000	10000
精确的位数	13.2	12.8	12.7

除了 Lanczos，还有其他近似方法。例如常见但精度极低的 Stirling 近似：

$$x!\sim\sqrt{2\pi x}\left(\frac{x}{e}\right)^x$$

另一种叫 *luschnyCF*[4] 的近似如下：

$$N = x + \frac{1}{2}$$

$$x! \sim \sqrt{2\pi} N^N e^{-N} \left(\frac{N}{N + \cfrac{\frac{1}{24}}{N + \cfrac{\frac{3}{80}}{N + \cfrac{18029}{45360} \cdot \cfrac{1}{N + \cfrac{6272051}{14869008 \ N}}}}} \right)^N$$

此公式随着 x 增大，精度升高。当 x>25 时可以选用此公式，或者利用 $x!=x(x-1)!$ 来递归计算。

x	100	1000	10000
精确的位数	21.5	30.5	39.5

由余元公式，不难发现

$$\Gamma(p)\Gamma(1-p)=\frac{\pi}{\sin p\pi}(0<p<1)$$

那么我们只需要计算 x>0.5 的 Gamma 即可。

Psi 函数的定义是：

$$\psi(x)=(\ln\Gamma(x))'=\frac{\Gamma(x)'}{\Gamma(x)}$$

使用 Psi 函数便能计算 Gamma 函数的导数。

不难发现

$$\psi(x)+\psi(1-x)=\frac{\pi}{\tan p\pi}$$

那么我们只需要计算 x>0.5 的 Psi 即可。

13.8.4 Zeta 函数

Zeta 函数是与 Riemann 猜想相关的重要函数。正偶数的 Zeta 函数值总和 π 有关。这里我们讲如何实现 Zeta 函数的数值计算。

注意到

$$\zeta(s)=\sum_{k=1}^{\infty}\frac{1}{k^s}\forall \mathrm{Re}(s)>1$$

我们便可以计算 $x>1$ 的 Zeta。当 $x=1$，返回∞。

但是不难发现，这级数对于接近 1 的 s，收敛地非常缓慢。例如计算 zeta(2)，其实际值为$\frac{\pi^2}{6}$:

级数迭代次数	值
100	1.63498390018
1000	1.64393456668
10000	1.64483407182
100000	1.64492406690
1000000	1.64493306685
10000000	1.64493396685
10000000000	1.64493406675

我们发现，zeta(2)需要迭代超过 10000000000 次才能得到较为准确的数值。

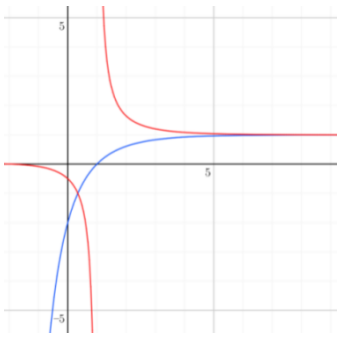
另外有

$$\zeta(s)=\frac{1}{1-2^{1-s}}\sum_{k=1}^{\infty}\frac{(-1)^{k-1}}{k^s}$$

$$\zeta(s)=2^s\pi^{s-1}\sin\frac{\pi s}{2}\Gamma(1-s)\zeta(1-s)$$

这样便可以计算$x\in\mathbb{R}$的 $\zeta(x)$。

另外，我们注意到 $x=1$ 是 $\zeta(x)$的无穷间断点，但取$\frac{1}{\zeta(x)}$便能消除无穷间断点。于是，我们便可以考察$\frac{1}{\zeta(x)}$的计算。



（zeta 函数（红）与 zeta 函数倒数（蓝）的图像）

这适合使用 Taylor 展开或者 Chebyshev 近似。SGC 使用的是 20 阶 Chebyshev 近似。

参考文献

①Lanczos 系数：https://www.mrob.com/pub/ries/lanczos-gamma.html

Coefficients for the Lanczos Approximation to the Gamma Function at MROB

②更多 Gamma 的近似公式：

http://www.luschny.de/math/factorial/approx/SimpleCases.html

Approximations for the Factorial Function

③Wolfram 数学天地（中文版）：https://mathworld.net.cn

https://mathworld.net.cn

④Numerical Recipes: *The Art of Scientific Computing* (Third Edition)

numerical.recipes/book.html

⑤Github: Scratch-VM

https://github.com/scratchfoundation/scratch-editor

scratchfoundation/scratch-editor: Scratch editor mono-repo

西江月 · 证明

即得易见平凡，仿照上例显然。留作习题答案略，读者自证不难。

反之亦然同理，推论自然成立。略去过程 Q.E.D.，由上可知证毕。