ENTERPRISE ARCHITECT

用户指南系列

脚本

Author: Sparx Systems

Date: 20/06/2023

Version: 16.1

创建于 ENTERPRISE ARCHITECT

目录

脚本

Enterprise Architect的脚本环境是一种灵活且易于使用的功能，它支持JavaScript和 Microsoft 脚本语言 JScript 和 VBScript。当任何脚本运行时，它都可以访问内置的 存储库" object 。 使用此脚本object ，您可以以编程方式检查和/或修改当前打开的模型中的元素。 Enterprise Architect还提供了特征的特性编辑器和工具来运行、调试和管理您的脚本。脚本是模块化的，可以使用*!include*指令按名称包含其他脚本。它们可以用于广泛的用途，从文档到验证和重构，它们可以在自动化耗时的任务方面提供巨大帮助。

脚本引擎支持

- Mozilla SpiderMonkey [版本1 .8]
- 微软脚本引擎

脚本语言

- JavaScript
- 脚本
- VBScript

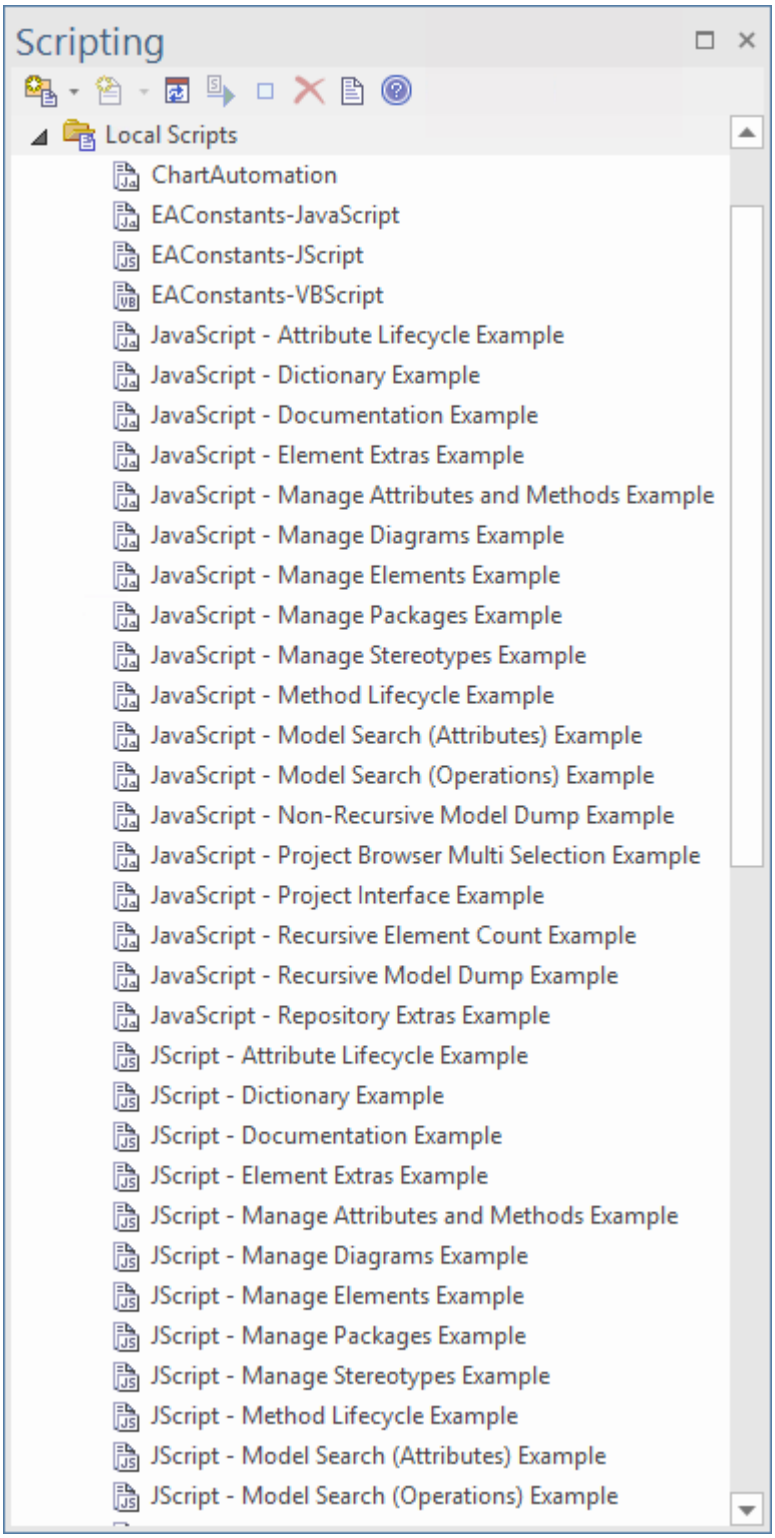
好处

- 对模型和元素组成进行检查和报告
- 修改和更新元素属性
- 运行查询以获取扩展模型信息
- 修改图表布局
- 从报告文档模板调用以填充报告
- 创建和实施流程工作流
- 包含在MDG 技术中以增强特定领域的语言
- 通过上下文菜单对脚本进行广泛的 UI 访问
- 进程内和进程外 COM 客户端的自动化服务器角色（脚本本身就是进程内客户端的一个示例；插件是另一个）
- 通过工作流安全进行元素访问治理
- 模型搜索集成

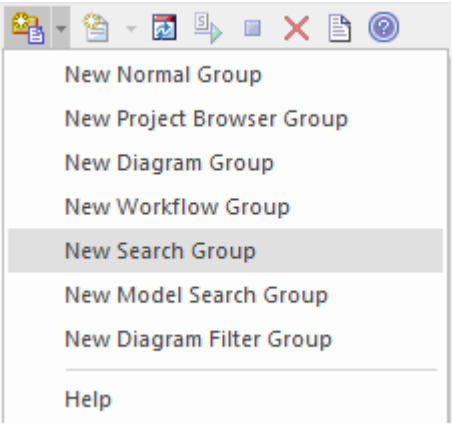
使用脚本

脚本的管理脚本是脚本窗口，显示了脚本树视图，可以用来审阅、创建和编辑脚本。

与基于文件并随Enterprise Architect安装的 Local其它不同，所有其他脚本都存储为模型资产，可以与模型的所有用户共享。脚本器可以帮助您进行脚本开发，脚本编辑器可以为您提供有关您可用的自动化接口的信息。您可以分析执行，例如通过记录脚本执行的序列图并暂停执行以查看局部变量。



脚本Groups



脚本在组中进行管理和包含。每个组都有一个称为 类型"的属性。此属性用于帮助Enterprise Architect决定脚本的使用方式和位置，以及应该从哪个特征中获得它。脚本组的属性可以从其快捷菜单中查看。

脚本储存

内置脚本是基于文件的，并与Enterprise Architect一起安装。它们出现在本地脚本组下。

您无法编辑或删除本地脚本，但您可以轻松地复制内容。

用户定义的脚本基于模型，因此可以由社区共享。它们列在它们所属的组中。

使用求解器

Enterprise Architect中Anywhere有JavaScript代码的地方，例如在仿真中，您现在可以使用名为 "Solver"（Solver类）的JavaScript构造与外部工具集成，并直接使用每个工具中的功能来简单直观地执行复杂的数学和图表功能。这些调用可帮助您轻松地在内置JavaScript引擎和每个环境之间交换变量。支持的两个数学库是 MATLAB 和 Octave。

要使用 Solver类，您需要了解首选数学库中可用的函数以及它们使用的参数，如产品文档中所述。

作为JavaScript引擎的一部分，Solver Classes 也可以立即被插件

访问插件

作家创建基于模型的JavaScript插件。

另请参阅*Octave Solver*、 *MATLAB Solver*和*Solvers*帮助主题。

注记

- 此功能在企业统一版和终极版中可用
- 如果您打算使用 Crossover/ WINE下的脚本功能，还必须安装 Internet Explorer 6.0 或以上版本

脚本窗口

脚本窗口由工具栏和按组显示所有脚本的视图组成。脚本组及其脚本也有上下文菜单，提供部分或全部这些选项：

- 组属性-在 脚本组属性”对话框中显示或编辑脚本组属性
- 运行脚本-执行选中的脚本（或者按住Ctrl的同时双击脚本名）
- 调试脚本-调试选中的脚本
- 编辑脚本-更新选中的脚本（或双击脚本名称显示 脚本编辑器”，通常显示脚本模板，由创建时分配的用户组类型或 脚本组”上的用户组类型属性对话）
- 重命名脚本——更改选中的组或脚本的名称
- 新建 VBScript/JScript/ JavaScript - 将新脚本添加到选定的用户组
- 导入工作流脚本- 显示 浏览器”对话框，您可以通过该对话框找到并选择要导入工作流脚本文件夹的工作流脚本源（.vbs）文件
- 删除Group/脚本-删除选中的用户组或脚本

您还可以将脚本从一个用户脚本文件夹移动或复制到另一个；至：

- 移动一个脚本，在脚本窗口中突出显示它，然后将它拖到它现在所属的用户脚本文件夹中
- 复制一个脚本，在脚本窗口中突出显示它，然后按住 Ctrl 键，同时将它拖到要复制它的用户脚本文件夹中

访问

功能区	特定>工具>脚本

脚本工具栏

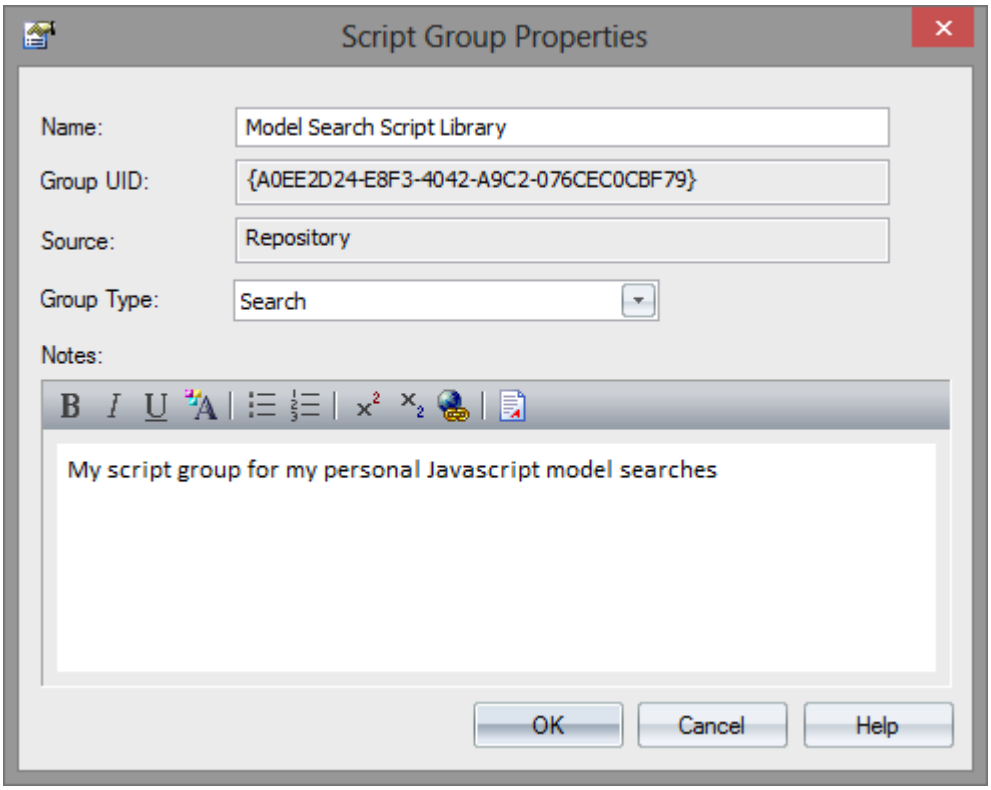
图标	行动
	创建一个新的脚本组；此选项显示您可以创建的脚本组类型的简短菜单，即： • 正常组 () • 浏览器窗口组 () •  图表 • 工作流组 () • 搜索组 () • 模型搜索组 新组将添加到脚本列表的末尾，并突出显示 新组”文本，以便您可以输入组名称。
	在选定的脚本组中新建一个脚本文件；这将显示您可以创建的脚本类型的简短菜单，即： • VBScript ()

	• JScript ( ) • JavaScript ( ) 新脚本将添加到所选组中列表的末尾，并突出显示 新脚本"文本，以便您可以输入脚本名称。
	刷新脚本窗口中的脚本树；此图标还会重新加载对工作流脚本所做的任何更改。
	编译并执行选定的脚本。 脚本的输出将写入系统输出窗口的 脚本"选项卡，您可以使用视图脚本输出显示该选项卡。
	停止正在执行的脚本；如果没有脚本正在执行，该图标将被禁用。
	从模型中删除脚本；您不能使用此图标删除脚本组（参见前面的 上下文菜单"项）、 本地脚本"组中的脚本或正在执行的脚本。 只有在 首选项"对话框的 常规"页面的 项目浏览器"面板中选中 确认删除"复选框时，系统才会提示您确认删除；如果未选择此选项，则不会显示任何提示。 脚本删除是永久性的 - 脚本无法恢复。
	显示系统输出窗口，并在 脚本"选项卡中显示最近执行的脚本的结果。

注记

- 此功能在企业统一版和终极版中可用
- 如果您添加、删除或更改脚本，您可能需要重新加载模型才能使更改生效
- 如果您选择删除包含脚本的脚本组，系统总是提示您确认该操作，而不管删除操作的任何系统设置；在确认删除之前确定您打算删除组及其脚本 - 脚本组和脚本的删除是永久性的

脚本组属性



当您创建脚本时，您在脚本组中开发它，其属性决定了该脚本如何提供给用户 - 通过浏览器窗口上下文菜单对特定类型的对象进行操作，或通过图表上下文菜单。使用脚本窗口工具栏上的第一个图标创建一个脚本组。

访问

功能区	特定>工具>脚本库>脚本>右键【组名】>组属性

定义脚本Group属性

字段/按钮	行动
名称	类型在脚本组的名称中。
组 UID	（只读）为组自动分配的GUID。
源	（只读）用于创建脚本的模板的位置。
组类型	单击下拉箭头并选择组中包含的脚本类型；这可以是以下之一： • Normal - ( ) 一般模型脚本 • 浏览器window - ( ) 在浏览器窗口'Scripts'上下文菜单选项中列出并可从

	其执行的脚本 • Workflow - ( ) 由Enterprise Architect的工作流引擎执行的脚本；您只能创建这种类型的 VB 脚本 • 搜索 - ( ) 可以作为模型搜索执行的脚本；这些脚本列在模型搜索窗口的"搜索"字段中，在列表的最后一个类别中 • 图表可以从图表上下文菜单的 'Scripts' 子菜单中执行的脚本 • 在项目中查找-( ) 可以从模型搜索视图中某个上下文菜单的"脚本"子菜单中执行的脚本，关于执行成功的结果# 包括 CLASSGUID 和 CLASSTYPE 的搜索，或查询构建的搜索 • 元素- 可以从元素上下文菜单的"脚本"子菜单中执行的脚本；可从浏览器窗口、图表、模型搜索、元素列表、包浏览器和甘特视图访问 • 包- 可以从包上下文菜单的"脚本"子菜单中执行的脚本；从浏览器窗口访问 • 图表- 可以从图表的 脚本"上下文菜单选项执行的脚本；可从浏览器窗口和图表访问 • 链接 - 可以从连接器的 脚本"上下文菜单选项执行的脚本；可从图表访问
注记	类型在您需要的有关此脚本组的任何注释中。

JavaScript Math Library

传奇的 Cephes 数学库与Enterprise Architect中可用的JavaScript引擎完全紧密集成。该库包含 400 多个用于科学和工程应用的高质量数学例程，为希望将其工程和系统模型提升到新水平的建模者提供了广泛的数学潜力。

函数库实现了 IEEE Std 754 双精度标准。

- Arithmetic and Algebraic
- Exponential and Trigonometric
- Exponential integral
- Gamma
- Error function
- Bessel
- Hypergeometric
- Elliptic
- Probability
- Miscellaneous
- Matrix
- Numerical Integration
- Complex Arithmetic
- Complex Exponential and Trigonometric
- errors

算法与代数

- sqrt -根
- lsqrt - 整数根
- cbrt - 立方体根
- polevl, p1evl - 计算多项式
- chbevl - 评估 Chebyshev 系列
- round - round到最接近的整数值
- ceil - 向上截断为整数
- floor - 向下截断为整数
- frexp - 提取指数
- ldexp - 将整数添加到指数
- fabs - 绝对值
- signbit - 将符号位返回为int
- isnan - 数字测试
- isfinite - 有限测试
- poladd - 添加多项式
- polsub - 多项式减法
- polmul - 多项式相乘
- poldiv - 多项式相除
- polsbt - 替代多项式变量
- poleva - 计算多项式
- polclr - 将所有系数设置为零
- polmov - 复制系数

sqrt

Square root.

SYNOPSIS:

double x, y, sqrt();
y = sqrt(x);

DESCRIPTION:

Returns the square root of x.

Range reduction involves isolating the power of two of the argument and using a polynomial approximation to obtain a rough value for the square root. Then Heron's iteration is used three times to converge to an accurate value.

ACCURACY:

		Relative error:		
arithmetic	domain	# trials	peak	rms
DEC	0, 10	60000	2.1e-17	7.9e-18
IEEE	0,1.7e308	30000	1.7e-16	6.3e-17

ERROR MESSAGES:

message	condition	value returned
domain	x < 0	0.0

lsqrt

Integer square root.

SYNOPSIS:

long x, y;

long lsqrt();

y = lsqrt(x);

DESCRIPTION:

Returns a long integer square root of the long integer argument. The computation is by binary long division. The largest possible result is lsqrt(2,147,483,647) = 46341.

If x < 0, the square root of |x| is returned, and an error message is available.

ACCURACY:

An extra, roundoff, bit is computed; hence the result is the nearest integer to the actual square root.

cbrt

Cube root.

SYNOPSIS:

double x, y, cbrt();
y = cbrt(x);

DESCRIPTION:

Returns the cube root of the argument, which could be negative. Range reduction involves determining the power of 2 of the argument. A polynomial of degree 2 applied to the mantissa, and multiplication by the cube root of 1, 2, or 4 approximates the root to within about 0.1%. Then Newton's iteration is used three times to converge to an accurate result.

ACCURACY:

	Relative error:			
arithmetic	domain	# trials	peak	rms
DEC	-10,10	200000	1.8e-17	6.2e-18
IEEE	0,1e308	30000	1.5e-16	5.0e-17

JavaScript:

```
//
//Plot of y = 3vx.
//

function plotYforX(x1, x2)
{
	for(var x = x1; x <= x2; x++)
	{
		var y = cephes.cbrt(x);
		Session.Output("plot of x for " + x + " gives y of " + y);
	}
}

function main()
{
	plotYforX(-1,6);
}
main();
```

polevl

Evaluate polynomial.

SYNOPSIS:

```
int N;
double x, y, coef[N+1], polevl[];
y = polevl(x, coef, N);
```

DESCRIPTION:

Evaluates polynomial of degree N:

$$y = C_0 + C_1 x + C_2 x^2 + \ldots + C_N x^N$$

Coefficients are stored in reverse order:

coef[0] = C_N , ..., coef[N] = C_0 .

The function p1evl() assumes that coef[N] = 1.0 and is omitted from the array. Its calling arguments are otherwise the same as polevl().

SPEED:

In the interest of speed, there are no checks for out of bounds arithmetic. This routine is used by most of the functions in the library. Depending on available equipment features, the user might want to rewrite the program in microcode or assembly language.

JavaScript:

Example:

```
function stirlingFormula(x)
{
    var STIR = [ 7.873113957930936 28397E-4, -2.29549961613378126380E-4,
            -2.68132617805781232825E-3, 3.47222221605458667310E-3,
             8.33333333333482257126E-2 ];
    var SQTPI = 2.50662827463100050242E0;
    var MAXSTIR = 143.01608;
    var w = 1.0 / x;
    var y = cephes.exp(x);
```

```
    var w = 1.0 + w * cephes.polevl(w, STIR, 4);
    if (x > MAXSTIR) {
        var v = cephes.pow(x, 0.5 * x - 0.25);
        y = v * (v / y);
    } else {
        y = cephes.pow(x, x - 0.5) / y;
    }
    y = SQTPI * y * w;
    return y;
}
```

chbevl

Evaluate Chebyshev series.

SYNOPSIS:

```
int N;
double x, y, coef[N], chebevl();

y = chbevl(x, coef, N);
```

DESCRIPTION:

Evaluates the series

$$y = \sum_{i=0}^{N-1} coef[i] \, T_i(x/2)$$

of Chebyshev polynomials Ti at argument x/2.

Coefficients are stored in reverse order, i.e. the zero order term is last in the array. Note N is the number of coefficients, not the order.

If coefficients are for the interval a to b, x must have been transformed to x -> 2(2x - b - a)/(b-a) before entering the routine. This maps x from (a, b) to (-1, 1), over which the Chebyshev polynomials are defined.

If the coefficients are for the inverted interval, in which (a, b) is mapped to (1/b, 1/a), the transformation required is x -> 2(2ab/x - b - a)/(b-a). If b is infinity, this becomes x -> 4a/x - 1.

SPEED:

Taking advantage of the recurrence properties of the Chebyshev polynomials, the routine requires one more addition per loop than evaluating a nested polynomial of the same degree.

JavaScript:

```
var y = cephes.chbevl(x, coef, N);
```

round

Round double to nearest or even integer valued double

SYNOPSIS:

double x, y, round();

y = round(x);

DESCRIPTION:

Returns the nearest integer to x as a double precision floating point result. If x ends in 0.5 exactly, the nearest even integer is chosen.

ACCURACY:

If x is greater than 1/(2*MACHEP), its closest machine representation is already an integer, so rounding does not change it.

floor

SYNOPSIS:

```
double floor(x);
 double x,y;
y = floor(x);
```

DESCRIPTION:

floor() returns the largest integer less than or equal to x. It truncates toward minus infinity.

ceil

SYNOPSIS:

```
double ceil(x);
 double x, y;
y = ceil(x);
```

DESCRIPTION:

ceil() returns the smallest integer greater than or equal to x. It truncates toward plus infinity.

frexp

Extract exponent.

SYNOPSIS:

```
double frexp(x, expnt);
 double x;
 int expnt;
y = frexp(x, &expnt);
```

DESCRIPTION:

frexp() extracts the exponent from x. It returns an integer power of two to expnt and the significand between 0.5 and 1 to y. Thus x = y * 2**expn.

ldexp

SYNOPSIS:

```
double ldexp(x,n);
 double x;
 int n;
y = ldexp(x, n);
```

DESCRIPTION:

ldexp() multiplies x by 2**n.

fabs

Absolute value.

SYNOPSIS:

double x, y;
y = fabs(x);

DESCRIPTION:

Returns the absolute value of the argument.

signbit

SYNOPSIS:

```
int signbit(x);
double x;
int n;
n = signbit(x);
```

DESCRIPTION:

signbit(x) returns 1 if the sign bit of x is 1, else 0.

isnan

SYNOPSIS:

```
int isnan(x);
double x;
int n;

n = isnan(x);
```

DESCRIPTION:

Returns true if x is not a number.

isfinite

SYNOPSIS:

```
int isfinite();
double x;
int n;

n = isfinite(x);
```

DESCRIPTION:

Return true if x is not infinite and is not a NaN

poladd

Polynomial Addition

SYNOPSIS:

```
int maxpol, na, nb, nc;
double a[na], b[nb], c[nc];

nc = max(na, nb);
polini( nc );
poladd( a, na, b, nb, c );
```

DESCRIPTION:

poladd(a, na, b, nb, c); c = b + a, nc = max(na, nb)

In this description a, b, c are polynomials of degree na, nb, nc respectively.

The degree of a polynomial cannot exceed a run-time value MAXPOL.

An operation that attempts to use or generate a polynomial of higher degree might produce a result that suffers truncation at degree MAXPOL.

The value of MAXPOL is set by calling the function

polini(MAXPOL);

Each polynomial is represented by an array containing its coefficients, together with a separately declared integer equal to the degree of the polynomial.

The coefficients appear in ascending order; that is,

$$a(x) = a[0] + a[1] * x + a[2] * x^2 + \ldots + a[na] * x^{na} .$$

polsub

Polynomial Subtraction

SYNOPSIS:

```
int maxpol, na, nb, nc;
double a[], b[], c[];

nc = max(na, nb);
polini( nc );
polsub( a, na, b, nb, c );
```

DESCRIPTION:

polsub(a, na, b, nb, c); c = b - a, nc = max(na, nb)

a, b, c are polynomials of degree na, nb, nc respectively.

The degree of a polynomial cannot exceed a run-time value MAXPOL.

An operation that attempts to use or generate a polynomial of higher degree might produce a result that suffers truncation at degree MAXPOL.

The value of MAXPOL is set by calling the function

polini(MAXPOL);

Each polynomial is represented by an array containing its coefficients, together with a separately declared integer equal to the degree of the polynomial.

The coefficients appear in ascending order; that is:

$$a(x) = a[0] + a[1] * x + a[2] * x^2 + \ldots + a[na] * x^{na} .$$

polmul

Polynomial Multiplication

SYNOPSIS:

```
int maxpol, na, nb, nc;
double a[], b[], c[];

nc = na + nb;
polini( nc );
polmul( a, na, b, nb, c );
```

DESCRIPTION:

polmul(a, na, b, nb, c); c = b * a, nc = na + nb

a, b, c are polynomials of degree na, nb, nc respectively.

The degree of a polynomial cannot exceed a run-time value MAXPOL.

An operation that attempts to use or generate a polynomial of higher degree might produce a result that suffers truncation at degree MAXPOL.

The value of MAXPOL is set by calling the function

polini(MAXPOL);

Each polynomial is represented by an array containing its coefficients, together with a separately declared integer equal to the degree of the polynomial.

The coefficients appear in ascending order; that is,

$$a(x) = a[0] + a[1] * x + a[2] * x^2 + \ldots + a[na] * x^{na} .$$

poldiv

Polynomial Division

SYNOPSIS:

```
int maxpol, na, nb, nc;
double a[], b[], c[];

nc = na + nb
polini( MAXPOL );
i = poldiv( a, na, b, nb, c );
```

DESCRIPTION:

i = poldiv(a, na, b, nb, c); c = b / a, nc = MAXPOL

returns i = the degree of the first nonzero coefficient of a.

The computed quotient c must be divided by x^i.

An error message is printed if a is identically zero.

a, b, c are polynomials of degree na, nb, nc respectively.

The degree of a polynomial cannot exceed a run-time value MAXPOL.

An operation that attempts to use or generate a polynomial of higher degree might produce a result that suffers truncation at degree MAXPOL.

The value of MAXPOL is set by calling the function

polini(MAXPOL);

Each polynomial is represented by an array containing its coefficients, together with a separately declared integer equal to the degree of the polynomial.

The coefficients appear in ascending order; that is,

$$a(x) = a[0] + a[1] * x + a[2] * x^2 + \ldots + a[na] * x^{na} .$$

polsbt

Substitute Polynomial Variable

SYNOPSIS:

int a, b;
double a[na], b[nb], c[nc];
polsbt(a, na, b, nb, c);

DESCRIPTION:

If a and b are polynomials, and t = a(x), then
c(t) = b(a(x))
is a polynomial found by substituting a(x) for t.

The subroutine call for this is:
polsbt(a, na, b, nb, c);

a, b, c are polynomials of degree na, nb, nc respectively.
The degree of a polynomial cannot exceed a run-time value MAXPOL.
An operation that attempts to use or generate a polynomial of higher degree might produce a result that suffers truncation at degree MAXPOL.
The value of MAXPOL is set by calling the function
polini(MAXPOL);

Each polynomial is represented by an array containing its coefficients, together with a separately declared integer equal to the degree of the polynomial.
The coefficients appear in ascending order; that is,

$$a(x) = a[0] + a[1] * x + a[2] * x^2 + ... + a[na] * x^{na} .$$

poleva

Polynomial Evaluation

SYNOPSIS:

```
int na;
double sum, x;
double a[na];

sum = poleva( a, na, x );
```

DESCRIPTION:

Evaluate polynomial a(t) at t = x.

The polynomial is represented by an array containing its coefficients, together with a separately declared integer equal to the degree of the polynomial.

The coefficients appear in ascending order; that is,

$$a(x) = a[0] + a[1] * x + a[2] * x^2 + ... + a[na] * x^{na} .$$

polclr

Clear Polynomial

SYNOPSIS:

```
int na;
double a[na];
polclr( a, na );
```

DESCRIPTION:

Set all coefficients of polynomial a to zero, up to a[na].

The polynomial is represented by an array containing its coefficients, together with a separately declared integer equal to the degree of the polynomial.

The coefficients appear in ascending order; that is,

$$a(x) = a[0] + a[1] * x + a[2] * x^2 + \ldots + a[na] * x^{na} .$$

polmov

Move Polynomial

SYNOPSIS:

```
int na;
double a[na], b[na];
polmov( a, na, b );
```

DESCRIPTION:

Set b = a. Copies coefficients of polynomial a, to b.

The polynomial is represented by an array containing its coefficients, together with a separately declared integer equal to the degree of the polynomial.

The coefficients appear in ascending order; that is,

$$a(x) = a[0] + a[1] * x + a[2] * x^2 + ... + a[na] * x^{na} .$$

指数和三角函数

- acos - 反cosine
- acosh - 反双曲cosine
- asinh - 反正弦双曲sine
- atanh - 弧双曲正切
- asin - 反正弦
- atan - 反正切
- atan2 - 象限正确反正切
- cos - 余弦
- cosdg - 以度为单位的 arg 的余弦
- exp - 以 e 为底的指数
- exp2 - 以 2 为底的指数
- exp10 - 以 10 为底的指数
- cosh - 双曲cosine
- sinh - 双曲sine
- tanh - 双曲正切
- log - 对数，以 e 为底
- log2 - 对数，以 2 为底
- log10 - 以 10 为底的对数
- pow - 电源
- powi -整数
- sin - Sine
- sindg - arg 的Sine ，以度为单位
- tan - 正切
- tandg - arg 的正切，以度为单位

acos

Inverse circular cosine.

SYNOPSIS:

double x, y, acos();
y = acos(x);

DESCRIPTION:

Returns radian angle between 0 and pi whose cosine is x.

Analytically, acos(x) = pi/2 - asin(x). However if |x| is near 1, there is cancellation error in subtracting asin(x) from pi/2. Hence if x < -0.5, acos(x) = pi - 2.0 * asin(sqrt((1+x)/2)); or if x > +0.5, acos(x) = 2.0 * asin(sqrt((1-x)/2)).

ACCURACY:

	Relative error:			
arithmetic	domain	# trials	peak	rms
DEC	-1, 1	50000	3.3e-17	8.2e-18
IEEE	-1, 1	10^6	2.2e-16	6.5e-17

ERROR MESSAGES:

message	condition	value returned
domain	|x| > 1	NAN

acosh

Inverse hyperbolic cosine.

SYNOPSIS:

```
double x, y, acosh();
y = acosh(x);
```

DESCRIPTION:

Returns the inverse hyperbolic cosine of an argument.

If 1 <= x < 1.5, a rational approximation:

sqrt(z) * P(z)/Q(z)

where z = x-1, is used. Otherwise:

acosh(x) = log(x + sqrt((x-1)(x+1)).

ACCURACY:

		Relative error:		
arithmetic	domain	# trials	peak	rms
DEC	1,3	30000	4.2e-17	1.1e-17
IEEE	1,3	30000	4.6e-16	8.7e-17

ERROR MESSAGES:

message	condition	value returned
domain	|x| < 1	NAN

asinh

Inverse hyperbolic sine.

SYNOPSIS:

double x, y, asinh();
y = asinh(x);

DESCRIPTION:

Returns the inverse hyperbolic sine of an argument.

If |x| < 0.5, the function is approximated by a rational form x + x**3 P(x)/Q(x).
Otherwise, asinh(x) = log(x + sqrt(1 + x*x)).

ACCURACY:

			Relative error:	
arithmetic	domain	# trials	peak	rms
DEC	-3,3	75000	4.6e-17	1.1e-17
IEEE	-1,1	30000	3.7e-16	7.8e-17
IEEE	1,3	30000	2.5e-16	6.7e-17

atanh

Inverse hyperbolic tangent.

SYNOPSIS:

double x, y, atanh();
y = atanh(x);

DESCRIPTION:

Returns the inverse hyperbolic tangent of an argument in the range MINLOG to MAXLOG.

If |x| < 0.5, the rational form x + x**3 P(x)/Q(x) is employed. Otherwise:
atanh(x) = 0.5 * log((1+x)/(1-x)).

ACCURACY:

		Relative error:		
arithmetic	domain	# trials	peak	rms
DEC	-1,1	50000	2.4e-17	6.4e-18
IEEE	-1,1	30000	1.9e-16	5.2e-17

asin

Inverse circular sine.

SYNOPSIS:

double x, y, asin();
y = asin(x);

DESCRIPTION:

Returns the radian angle between -pi/2 and +pi/2 whose sine is x.

A rational function of the form x + x**3 P(x**2)/Q(x**2) is used for |x| in the interval [0, 0.5]. If |x| > 0.5 it is transformed by the identity:

asin(x) = pi/2 - 2 asin(sqrt((1-x)/2)).

ACCURACY:

		Relative error:		
arithmetic	domain	# trials	peak	rms
DEC	-1, 1	40000	2.6e-17	7.1e-18
IEEE	-1, 1	10^6	1.9e-16	5.4e-17

ERROR MESSAGES:

message	condition	value returned
domain	|x| > 1	NAN

atan

Inverse circular tangent (arctangent).

SYNOPSIS:

double x, y, atan();
y = atan(x);

DESCRIPTION:

Returns the radian angle between -pi/2 and +pi/2 whose tangent is x.

Range reduction is from three intervals into the interval from zero to 0.66. The approximant uses a rational function of degree 4/5 of the form x + x**3 P(x)/Q(x).

ACCURACY:

Relative error:

arithmetic	domain	# trials	peak	rms
DEC	-10, 10	50000	2.4e-17	8.3e-18
IEEE	-10, 10	10^6	1.8e-16	5.0e-17

atan2

Quadrant correct inverse circular tangent.

SYNOPSIS:

double x, y, z, atan2();
z = atan2(y, x);

DESCRIPTION:

Returns the radian angle whose tangent is y/x.
Define compile time symbol ANSIC = 1 for ANSI standard, range -PI < z <= +PI, args (y,x);
else ANSIC = 0 for range 0 to 2PI, args (x,y).

ACCURACY:

	Relative error:			
arithmetic	domain	# trials	peak	rms
IEEE	-10, 10	10^6	2.5e-16	6.9e-17

cos

Circular cosine.

SYNOPSIS:

```
double x, y, cos();
y = cos(x);
```

DESCRIPTION:

Range reduction is into intervals of pi/4. The reduction error is nearly eliminated by contriving an extended precision modular arithmetic.

Two polynomial approximating functions are employed.
Between 0 and pi/4 the cosine is approximated by:
    1 - x**2 Q(x**2).
Between pi/4 and pi/2 the sine is represented as:
    x + x**3 P(x**2).

ACCURACY:

		Relative error:		
arithmetic	domain	# trials	peak	rms
IEEE	-1.07e9,+1.07e9	130000	2.1e-16	5.4e-17
DEC	0,+1.07e9	17000	3.0e-17	7.2e-18

cosdg

Circular cosine of angle in degrees.

SYNOPSIS:

```
double x, y, cosdg();
y = cosdg(x);
```

DESCRIPTION:

Range reduction is into intervals of 45 degrees. Two polynomial approximating functions are employed.
Between 0 and pi/4 the cosine is approximated by:

```
    1  -  x**2 P(x**2).
```

Between pi/4 and pi/2 the sine is represented as:

```
    x  +  x**3 P(x**2).
```

ACCURACY:

	Relative error:			
arithmetic	domain	# trials	peak	rms
DEC	+-1000	3400	3.5e-17	9.1e-18
IEEE	+-1000	30000	2.1e-16	5.7e-17

exp

Exponential function.

SYNOPSIS:

double x, y, exp();
y = exp(x);

DESCRIPTION:

Returns e (2.71828...) raised to the x power.

Range reduction is accomplished by separating the argument into an integer k and fraction f such that:

$$e^{x} = 2^{k} e^{f}$$

A Pade' form
1 + 2x P(x**2)/(Q(x**2) - P(x**2)) of degree 2/3 is used to approximate exp(f) in the basic interval [-0.5, 0.5].

ACCURACY:

	Relative error:			
arithmetic	domain	# trials	peak	rms
DEC	+- 88	50000	2.8e-17	7.0e-18
IEEE	+- 708	40000	2.0e-16	5.6e-17

Error amplification in the exponential function can be a serious matter. The error propagation involves:
exp(X(1+delta)) = exp(X) (1 + X*delta + ...)
This shows that a 1 lsb error in representing X produces a relative error of X times 1 lsb in the function. While the routine gives an accurate result for arguments that are exactly represented by a double precision computer number, the result contains an amplified roundoff error for large arguments not exactly represented.

ERROR MESSAGES:

message	condition	value returned
underflow	x < MINLOG	0.0
overflow	x > MAXLOG	INFINITY

exp2

Base 2 exponential function.

SYNOPSIS:

double x, y, exp2();
y = exp2(x);

DESCRIPTION:

Returns 2 raised to the x power.

Range reduction is accomplished by separating the argument into an integer k and fraction f, such that:

$$2^x = 2^k \, 2^f$$

A Pade' form:

1 + 2x P(x**2) / (Q(x**2) - x P(x**2))

approximates 2**x in the basic range [-0.5, 0.5].

ACCURACY:

		Relative error:		
arithmetic	domain	# trials	peak	rms
IEEE	-1022,+1024	30000	1.8e-16	5.4e-17

ERROR MESSAGES:

message	condition	value returned
underflow	x < -MAXL2	0.0
overflow	x > MAXL2	MAXNUM

For DEC arithmetic, MAXL2 = 127.
For IEEE arithmetic, MAXL2 = 1024.

exp10

Base 10 exponential function. (Common antilogarithm.)

SYNOPSIS:

```
double x, y, exp10();
y = exp10(x);
```

DESCRIPTION:

Returns 10 raised to the x power.

Range reduction is accomplished by expressing the argument as 10**x = 2**n 10**f, with |f| < 0.5 log10(2).
The Pade' form:

```
   1 + 2x P(x**2)/(Q(x**2) - P(x**2))
```

is used to approximate 10**f.

ACCURACY:

		Relative error:		
arithmetic	domain	# trials	peak	rms
IEEE	-307,+307	30000	2.2e-16	5.5e-17

Test result from an earlier version (2.1):

DEC	-38,+38	70000	3.1e-17	7.0e-18

ERROR MESSAGES:

message	condition	value returned
underflow	x < -MAXL10	0.0
overflow	x > MAXL10	MAXNUM

DEC arithmetic: MAXL10 = 38.230809449325611792.
IEEE arithmetic: MAXL10 = 308.2547155599167.

cosh

Hyperbolic cosine.

SYNOPSIS:

double x, y, cosh();
y = cosh(x);

DESCRIPTION:

Returns the hyperbolic cosine of an argument in the range MINLOG to MAXLOG.

cosh(x) = (exp(x) + exp(-x))/2.

ACCURACY:

		Relative error:		
arithmetic	domain	# trials	peak	rms
DEC	+- 88	50000	4.0e-17	7.7e-18
IEEE	+-MAXLOG	30000	2.6e-16	5.7e-17

ERROR MESSAGES:

message	condition	value returned
overflow	|x| > MAXLOG	MAXNUM

sinh

Hyperbolic sine.

SYNOPSIS:

double x, y, sinh();
y = sinh(x);

DESCRIPTION:

Returns the hyperbolic sine of an argument in the range MINLOG to MAXLOG.

The range is partitioned into two segments. If |x| <= 1, a rational function of the form x + x**3 P(x)/Q(x) is employed. Otherwise the calculation is sinh(x) = (exp(x) - exp(-x))/2.

ACCURACY:

			Relative error:	
arithmetic	domain	# trials	peak	rms
DEC	+- 88	50000	4.0e-17	7.7e-18
IEEE	+-MAXLOG	30000	2.6e-16	5.7e-17

tanh

Hyperbolic tangent.

SYNOPSIS:

double x, y, tanh();
y = tanh(x);

DESCRIPTION:

Returns the hyperbolic tangent of an argument in the range MINLOG to MAXLOG.

A rational function is used for |x| < 0.625. The form:
x + x**3 P(x)/Q(x) of Cody _& Waite
is employed.
Otherwise:
tanh(x) = sinh(x)/cosh(x) = 1 - 2/(exp(2x) + 1).

ACCURACY:

		Relative error:		
arithmetic	domain	# trials	peak	rms
DEC	-2,2	50000	3.3e-17	6.4e-18
IEEE	-2,2	30000	2.5e-16	5.8e-17

log

Natural logarithm.

SYNOPSIS:

double x, y, log();
y = log(x);

DESCRIPTION:

Returns the base e (2.718...) logarithm of x.

The argument is separated into its exponent and fractional parts. If the exponent is between -1 and +1, the logarithm of the fraction is approximated by:

log(1+x) = x - 0.5 x**2 + x**3 P(x)/Q(x).

Otherwise, setting z = 2(x-1)/x+1),

log(x) = z + z**3 P(z)/Q(z).

ACCURACY:

	Relative error:			
arithmetic	domain	# trials	peak	rms
IEEE	0.5, 2.0	150000	1.44e-16	5.06e-17
IEEE	+-MAXNUM	30000	1.20e-16	4.78e-17
DEC	0, 10	170000	1.8e-17	6.3e-18

In the tests over the interval [+-MAXNUM], the logarithms of the random arguments were uniformly distributed over [0,MAXLOG].

ERROR MESSAGES:

singularity: x = 0; returns -INFINITY
domain: x < 0; returns NAN

log2

Base 2 logarithm.

SYNOPSIS:

double x, y, log2();
y = log2(x);

DESCRIPTION:

Returns the base 2 logarithm of x.

The argument is separated into its exponent and fractional parts. If the exponent is between -1 and +1, the base e logarithm of the fraction is approximated by:

log(1+x) = x - 0.5 x**2 + x**3 P(x)/Q(x).

Otherwise, setting z = 2(x-1)/x+1),

log(x) = z + z**3 P(z)/Q(z).

ACCURACY:

	Relative error:			
arithmetic	domain	# trials	peak	rms
IEEE	0.5, 2.0	30000	2.0e-16	5.5e-17
IEEE	exp(+-700)	40000	1.3e-16	4.6e-17

In the tests over the interval [exp(+-700)], the logarithms of the random arguments were uniformly distributed.

ERROR MESSAGES:

singularity: x = 0; returns -INFINITY
domain: x < 0; returns NAN

log10

Common logarithm.

SYNOPSIS:

double x, y, log10();
y = log10(x);

DESCRIPTION:

Returns logarithm to the base 10 of x.

The argument is separated into its exponent and fractional parts. The logarithm of the fraction is approximated by:

log(1+x) = x - 0.5 x**2 + x**3 P(x)/Q(x).

ACCURACY:

		Relative error:		
arithmetic	domain	# trials	peak	rms
IEEE	0.5, 2.0	30000	1.5e-16	5.0e-17
IEEE	0, MAXNUM	30000	1.4e-16	4.8e-17
DEC	1, MAXNUM	50000	2.5e-17	6.0e-18

In the tests over the interval [1, MAXNUM], the logarithms of the random arguments were uniformly distributed over [0, MAXLOG].

ERROR MESSAGES:

singularity: x = 0; returns -INFINITY
domain: x < 0; returns NAN

pow

Power function

SYNOPSIS:

double x, y, z, pow();
z = pow(x, y);

DESCRIPTION:

Computes x raised to the yth power. Analytically:

x**y = exp(y log(x)).

Following Cody and Waite, this program uses a lookup table of 2**-i/16 and pseudo extended precision arithmetic to obtain an extra three bits of accuracy in both the logarithm and the exponential.

ACCURACY:

		Relative error:		
arithmetic	domain	# trials	peak	rms
IEEE	-26,26	30000	4.2e-16	7.7e-17
DEC	-26,26	60000	4.8e-17	9.1e-18

1/26 < x < 26, with log(x) uniformly distributed.
-26 < y < 26, y uniformly distributed.

IEEE	0,8700	30000	1.5e-14	2.1e-15

0.99 < x < 1.01, 0 < y < 8700, uniformly distributed.

ERROR MESSAGES:

message	condition	value returned
overflow	x**y > MAXNUM	INFINITY
underflow	x**y < 1/MAXNUM	0.0
domain	x<0 and y noninteger	0.0

powi

Real raised to integer power.

SYNOPSIS:

double x, y, powi();
int n;

y = powi(x, n);

DESCRIPTION:

Returns an argument x raised to the nth power. The routine efficiently decomposes n as a sum of powers of two. The desired power is a product of two-to-the-kth powers of x. Thus to compute the 32767 power of x requires 28 multiplications instead of 32767 multiplications.

ACCURACY:

	Relative error:				
arithmetic	x domain	n domain	# trials	peak	rms
DEC	.04,26	-26,26	100000	2.7e-16	4.3e-17
IEEE	.04,26	-26,26	50000	2.0e-15	3.8e-16
IEEE	1,2	-1022,1023	50000	8.6e-14	1.6e-14

Returns MAXNUM on overflow, zero on underflow.

sin

Circular sine.

SYNOPSIS:

double x, y, sin();
y = sin(x);

DESCRIPTION:

Range reduction is into intervals of pi/4. The reduction error is nearly eliminated by contriving an extended precision modular arithmetic.

Two polynomial approximating functions are employed.
Between 0 and pi/4 the sine is approximated by:

x + x**3 P(x**2).

Between pi/4 and pi/2 the cosine is represented as:
1 - x**2 Q(x**2).

ACCURACY:

		Relative error:		
arithmetic	domain	# trials	peak	rms
DEC	0, 10	150000	3.0e-17	7.8e-18
IEEE	-1.07e9,+1.07e9	130000	2.1e-16	5.4e-17

ERROR MESSAGES:

message	condition	value returned
total loss	x > 1.073741824e9	0.0

Partial loss of accuracy begins to occur at x = 2**30 = 1.074e9. The loss is not gradual, but jumps suddenly to about 1 part in 10e7. Results might be meaningless for x > 2**49 = 5.6e14. The routine as implemented flags a TLOSS error for x > 2**30 and returns 0.0.

sindg

Circular sine of an angle in degrees.

SYNOPSIS:

double x, y, sindg();
y = sindg(x);

DESCRIPTION:

Range reduction is into intervals of 45 degrees. Two polynomial approximating functions are employed.
Between 0 and pi/4 the sine is approximated by:
x + x**3 P(x**2).

Between pi/4 and pi/2 the cosine is represented as:
1 - x**2 P(x**2).

ACCURACY:

		Relative error:		
arithmetic	domain	# trials	peak	rms
DEC	+-1000	3100	3.3e-17	9.0e-18
IEEE	+-1000	30000	2.3e-16	5.6e-17

ERROR MESSAGES:

message	condition	value returned
total loss	x > 8.0e14 (DEC)	0.0
	x > 1.0e14 (IEEE)	

tan

Circular tangent.

SYNOPSIS:

double x, y, tan();
y = tan(x);

DESCRIPTION:

Returns the circular tangent of the radian argument x.
Range reduction is modulo pi/4.
A rational function:

x + x**3 P(x**2)/Q(x**2)

is employed in the basic interval [0, pi/4].

ACCURACY:

	Relative error:			
arithmetic	domain	# trials	peak	rms
DEC	+-1.07e9	44000	4.1e-17	1.0e-17
IEEE	+-1.07e9	30000	2.9e-16	8.1e-17

ERROR MESSAGES:

message	condition	value returned
total loss	x > 1.073741824e9	0.0

tandg

Circular tangent of argument in degrees.

SYNOPSIS:

```
double x, y, tandg();
y = tandg(x);
```

DESCRIPTION:

Returns the circular tangent of the argument x in degrees.
Range reduction is modulo pi/4. A rational function:

```
x + x**3 P(x**2)/Q(x**2)
```

is employed in the basic interval [0, pi/4].

ACCURACY:

			Relative error:	
arithmetic	domain	# trials	peak	rms
DEC	0,10	8000	3.4e-17	1.2e-17
IEEE	0,10	30000	3.2e-16	8.4e-17

ERROR MESSAGES:

message	condition	value returned
total loss	x > 8.0e14 (DEC) x > 1.0e14 (IEEE)	0.0
singularity	x = 180 k + 90	MAXNUM

指数积分

- expn - 指数积分
- shichi - 双曲sine和cosine积分
- sici - Sine和cosine积分

expn

Exponential integral En.

SYNOPSIS:

```
int n;
double x, y, expn();
y = expn(n, x);
```

DESCRIPTION:

Evaluates the exponential integral.

$$E_n(x) = \int_1^{\infty} \frac{e^{-xt}}{t^n}\, dt.$$

Both n and x must be nonnegative.

The routine employs either a power series, a continued fraction, or an asymptotic formula depending on the relative values of n and x.

ACCURACY:

		Relative error:		
arithmetic	domain	# trials	peak	rms
DEC	0, 30	5000	2.0e-16	4.6e-17
IEEE	0, 30	10000	1.7e-15	3.6e-16

shichi

Hyperbolic sine and cosine integrals.

SYNOPSIS:

```
double x, Chi, Shi, shichi();
shichi(x, &Chi, &Shi);
```

DESCRIPTION:

Approximates the integrals

$$\mathrm{Chi}(x) = \mathrm{eul} + \ln x + \int_0^x \frac{\cosh t - 1}{t}\,dt,$$

$$\mathrm{Shi}(x) = \int_0^x \frac{\sinh t}{t}\,dt$$

where eul = 0.57721566490153286061 is Euler's constant. The integrals are evaluated by power series for x < 8 and by Chebyshev expansions for x between 8 and 88. For large x, both functions approach exp(x)/2x. Arguments greater than 88 in magnitude return MAXNUM.

ACCURACY:

Test interval 0 to 88.

			Relative error:	
arithmetic	function	# trials	peak	rms
DEC	Shi	3000	9.1e-17	
IEEE	Shi	30000	6.9e-16	1.6e-16

Absolute error, except relative when |Chi| > 1:

DEC	Chi	2500	9.3e-17	
IEEE	Chi	30000	8.4e-16	1.4e-16

sici

Sine and cosine integrals.

SYNOPSIS:

```
double x, Ci, Si, sici();
sici(x, &Si, &Ci);
```

DESCRIPTION:

Evaluates the integrals:

$$\mathrm{Ci}(x) = \mathrm{eul} + \ln x + \int_0^x \frac{\cos t - 1}{t}\,dt,$$

$$\mathrm{Si}(x) = \int_0^x \frac{\sin t}{t}\,dt$$

where eul = 0.57721566490153286061 is Euler's constant. The integrals are approximated by rational functions. For x > 8 auxiliary functions f(x) and g(x) are employed such that

```
Ci(x) = f(x) sin(x) - g(x) cos(x)
Si(x) = pi/2 - f(x) cos(x) - g(x) sin(x)
```

ACCURACY:

Test interval = [0,50].
Absolute error, except relative when > 1:

arithmetic function # trials peak rms

IEEE	Si	30000	4.4e-16	7.3e-17
IEEE	Ci	30000	6.9e-16	5.1e-17
DEC	Si	5000	4.4e-17	9.0e-18
DEC	Ci	5300	7.9e-17	5.2e-18

Gamma

- beta - beta
- lbeta - beta的自然log
- fac
- gamma - gamma
- lgam - gamma函数的对数
- incbet - 不完全beta积分
- incbi - 不完全beta积分的逆
- igam - 不完全gamma积分
- igamc - 补充gamma积分
- igami - 逆gamma积分
- psi - Psi (digamma)函数
- rgamma - 倒数Gamma

beta

Beta function.

SYNOPSIS:

```
double a, b, y, beta();
y = beta(a, b);
```

DESCRIPTION:

```
                 -     -
                | (a) | (b)
beta(a, b)  =  -----------
                 -
                | (a+b)
```

For large arguments the logarithm of the function is evaluated using lgam(), then exponentiated.

ACCURACY:

	Relative error:			
arithmetic	domain	# trials	peak	rms
DEC	0,30	1700	7.7e-15	1.5e-15
IEEE	0,30	30000	8.1e-14	1.1e-14

ERROR MESSAGES:

message	condition	value returned
overflow	log(beta) > MAXLOG	0.0
	a or b <0 integer	0.0

lbeta

Natural log of |beta|.

Return the sign of beta in sgngam.

fac

Factorial function.

SYNOPSIS:

```
double y, fac();
int i;
y = fac(i);
```

DESCRIPTION:

Returns factorial of i = 1 * 2 * 3 * ... * i.
fac(0) = 1.0.

Due to machine arithmetic bounds the largest value of i accepted is 33 in DEC arithmetic or 170 in IEEE arithmetic. Greater values, or negative ones, produce an error message and return MAXNUM.

ACCURACY:

For i < 34 the values are simply tabulated, and have full machine accuracy. If i > 55, fac(i) = gamma(i+1);

Relative error:

arithmetic	domain	peak
IEEE	0, 170	1.4e-15
DEC	0, 33	1.4e-17

gamma

Gamma function.

SYNOPSIS:

double x, y, gamma();
y = gamma(x);

DESCRIPTION:

Returns the gamma function of the argument. The result is correctly signed, and the sign (+1 or -1) is also returned in a global (extern) variable named sgngam. This variable is also filled in by the logarithmic gamma function lgam().

Arguments |x| <= 34 are reduced by recurrence and the function approximated by a rational function of degree 6/7 in the interval (2,3). Large arguments are handled by Stirling's formula. Large negative arguments are made positive using a reflection formula.

ACCURACY:

	Relative error:			
arithmetic	domain	# trials	peak	rms
DEC	-34, 34	10000	1.3e-16	2.5e-17
IEEE	-170,-33	20000	2.3e-15	3.3e-16
IEEE	-33, 33	20000	9.4e-16	2.2e-16
IEEE	33, 171.6	20000	2.3e-15	3.2e-16

Error for arguments outside the test range will be larger owing to error amplification by the exponential function.

lgam

Natural logarithm of gamma function.

SYNOPSIS:

double x, y, lgam();
y = lgam(x);

DESCRIPTION:

Returns the base e (2.718...) logarithm of the absolute value of the gamma function of the argument. The sign (+1 or -1) of the gamma function is returned in a global (extern) variable named sgngam.

For arguments greater than 13, the logarithm of the gamma function is approximated by the logarithmic version of Stirling's formula using a polynomial approximation of degree 4. Arguments between -33 and +33 are reduced by recurrence to the interval [2,3] of a rational approximation. The cosecant reflection formula is employed for arguments less than -33.

Arguments greater than MAXLGM return MAXNUM and an error message.

MAXLGM = 2.035093e36 for DEC arithmetic or 2.556348e305 for IEEE arithmetic.

ACCURACY:

arithmetic	domain	# trials	peak	rms
DEC	0, 3	7000	5.2e-17	1.3e-17
DEC	2.718, 2.035e36	5000	3.9e-17	9.9e-18
IEEE	0, 3	28000	5.4e-16	1.1e-16
IEEE	2.718, 2.556e305	40000	3.5e-16	8.3e-17

The error criterion was relative when the function magnitude was greater than one but absolute when it was less than one.

This test used the relative error criterion, though at certain points the relative error could be much higher than indicated.

IEEE	-200, -4	10000	4.8e-16	1.3e-16

incbet

Incomplete beta integral.

SYNOPSIS:

```
double a, b, x, y, incbet();
y = incbet(a, b, x);
```

DESCRIPTION:

Returns the incomplete beta integral of the arguments, evaluated from zero to x. The function is defined as:

```
                 x
    -            -
   | (a+b)      | |  a-1     b-1
 -----------    |   t   (1-t)   dt.
  -    -     | |
 | (a) | (b)   -
               0
```

The domain of definition is 0 <= x <= 1. In this implementation a and b are restricted to positive values. The integral from x to 1 can be obtained by the symmetry relation:

1 - incbet(a, b, x) = incbet(b, a, 1-x).

The integral is evaluated by a continued fraction expansion or, when b*x is small, by a power series.

ACCURACY:

Tested at uniformly distributed random points (a,b,x) with a and b in "domain" and x between 0 and 1.

			Relative error	
arithmetic	domain	# trials	peak	rms
IEEE	0,5	10000	6.9e-15	4.5e-16
IEEE	0,85	250000	2.2e-13	1.7e-14
IEEE	0,1000	30000	5.3e-12	6.3e-13
IEEE	0,10000	250000	9.3e-11	7.1e-12
IEEE	0,100000	10000	8.7e-10	4.8e-11

Outputs smaller than the IEEE gradual underflow threshold were excluded from these statistics.

ERROR MESSAGES:

message	condition	value returned
domain	x<0, x>1	0.0

underflow	0.0

incbi

Inverse of incomplete beta integral.

SYNOPSIS:

double a, b, x, y, incbi();
x = incbi(a, b, y);

DESCRIPTION:

Given y, the function finds x such that:

incbet(a, b, x) = y .

The routine performs interval halving or Newton iterations to find the root of incbet(a,b,x) - y = 0.

ACCURACY:

Relative error:

	x	a,b			
arithmetic	domain	domain	# trials	peak	rms
IEEE	0,1	.5,10000	50000	5.8e-12	1.3e-13
IEEE	0,1	.25,100	100000	1.8e-13	3.9e-15
IEEE	0,1	0,5	50000	1.1e-12	5.5e-15
VAX	0,1	.5,100	25000	3.5e-14	1.1e-15

With a and b constrained to half-integer or integer values:

IEEE	0,1	.5,10000	50000	5.8e-12	1.1e-13
IEEE	0,1	.5,100	100000	1.7e-14	7.9e-16

With a = .5, b constrained to half-integer or integer values:

IEEE	0,1	.5,10000	10000	8.3e-11	1.0e-11

igam

Incomplete gamma integral.

SYNOPSIS:

```
double a, x, y, igam();
y = igam(a, x);
```

DESCRIPTION:

The function is defined by

$$\mathrm{igam}(a,x) = \frac{1}{\Gamma(a)} \int_0^x e^{-t} t^{a-1}\, dt.$$

In this implementation both arguments must be positive. The integral is evaluated by either a power series or continued fraction expansion, depending on the relative values of a and x.

ACCURACY:

Relative error:

arithmetic	domain	# trials	peak	rms
IEEE	0,30	200000	3.6e-14	2.9e-15
IEEE	0,100	300000	9.9e-14	1.5e-14

igamc

Complemented incomplete gamma integral.

SYNOPSIS:

```
double a, x, y, igamc();
y = igamc(a, x);
```

DESCRIPTION:

The function is defined by

```
 igamc(a,x)   =   1 - igam(a,x)

                    inf.
                      -
           1       | |  -t  a-1
     =   -----     |   e   t    dt
         -     | |
        | (a)    -
                 x
```

In this implementation both arguments must be positive. The integral is evaluated by either a power series or continued fraction expansion, depending on the relative values of a and x.

ACCURACY:

Tested at random a, x.

	a	x	Relative error:		
arithmetic	domain	domain	# trials	peak	rms
IEEE	0.5,100	0,100	200000	1.9e-14	1.7e-15
IEEE	0.01,0.5	0,100	200000	1.4e-13	1.6e-15

igami

Inverse of complemented incomplete gamma integral.

SYNOPSIS:

double a, x, p, igami();
x = igami(a, p);

DESCRIPTION:

Given p, the function finds x such that

igamc(a, x) = p.

Starting with the approximate value

$$x = a\,t^3$$

where

$$t = 1 - d - \mathrm{ndtri}(p)\sqrt{d}$$

and

$$d = 1/9a,$$

the routine performs up to 10 Newton iterations to find the root of igamc(a,x) - p = 0.

ACCURACY:

Tested at random a, p in the intervals indicated.

	a	p		Relative error:	
arithmetic	domain	domain	# trials	peak	rms
IEEE	0.5,100	0,0.5	100000	1.0e-14	1.7e-15
IEEE	0.01,0.5	0,0.5	100000	9.0e-14	3.4e-15
IEEE	0.5,10000	0,0.5	20000	2.3e-13	3.8e-14

psi

Psi (digamma) function.

SYNOPSIS:

double x, y, psi();
y = psi(x);

DESCRIPTION:

$$\text{psi}(x) = \frac{d}{dx} \ln | (x)$$

is the logarithmic derivative of the gamma function.
For integer x:

$$\text{psi}(n) = -\text{EUL} + \sum_{k=1}^{n-1} 1/k.$$

This formula is used for 0 < n <= 10. If x is negative, it is transformed to a positive argument by the reflection formula psi(1-x) = psi(x) + pi cot(pi x). For general positive x, the argument is made greater than 10 using the recurrence psi(x+1) = psi(x) + 1/x. Then this asymptotic expansion is applied:

$$\text{psi}(x) = \log(x) - 1/2x - \sum_{k=1}^{\infty} \frac{B_{2k}}{2k\, x^{2k}}$$

where the B2k are Bernoulli numbers.

ACCURACY:

Relative error (except absolute when |psi| < 1):

arithmetic	domain	# trials	peak	rms
DEC	0,30	2500	1.7e-16	2.0e-17
IEEE	0,30	30000	1.3e-15	1.4e-16

IEEE -30,0 40000 1.5e-15 2.2e-16

ERROR MESSAGES:

message	condition	value returned
singularity	x integer <=0	MAXNUM

rgamma

Reciprocal gamma function.

SYNOPSIS:

double x, y, rgamma();

y = rgamma(x);

DESCRIPTION:

Returns one divided by the gamma function of the argument.

The function is approximated by a Chebyshev expansion in the interval [0,1]. Range reduction is by recurrence for arguments between -34.034 and +34.84425627277176174. 1/MAXNUM is returned for positive arguments outside this range. For arguments less than -34.034 the cosecant reflection formula is applied; logarithms are employed to avoid unnecessary overflow.

The reciprocal gamma function has no singularities, but overflow and underflow could occur for large arguments. These conditions return either MAXNUM or 1/MAXNUM with the appropriate sign.

ACCURACY:

	Relative error:			
arithmetic	domain	# trials	peak	rms
DEC	-30,+30	4000	1.2e-16	1.8e-17
IEEE	-30,+30	30000	1.1e-15	2.0e-16

For arguments less than -34.034 the peak error is in the order of 5e-15 (DEC), excepting overflow or underflow.

错误函数

- erf - 错误函数
- erfc - 补错函数
- 道dawsn - 道森积分
- fresnl - 菲涅耳积分

erf

Error function.

SYNOPSIS:

double x, y, erf();
y = erf(x);

DESCRIPTION:

The integral is

$$\mathrm{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x \exp(-t^2)\, dt.$$

The magnitude of x is limited to 9.231948545 for DEC arithmetic; 1 or -1 is returned outside this range.
For 0 <= |x| < 1, erf(x) = x * P4(x**2)/Q5(x**2); otherwise erf(x) = 1 - erfc(x).

ACCURACY:

		Relative error:		
arithmetic	domain	# trials	peak	rms
DEC	0,1	14000	4.7e-17	1.5e-17
IEEE	0,1	30000	3.7e-16	1.0e-16

erfc

Complementary error function.

SYNOPSIS:

```
double x, y, erfc();
y = erfc(x);
```

DESCRIPTION:

```
 1 - erf(x) =
                     inf.
                      -
             2       | |         2
  erfc(x)  =  --------    |    exp(- t) dt
           sqrt(pi)   | |
                      -
                      x
```

For small x, erfc(x) = 1 - erf(x); otherwise rational approximations are computed.
A special function expx2.c is used to suppress error amplification in computing exp(-x^2).

ACCURACY:

	Relative error:			
arithmetic	domain	# trials	peak	rms
IEEE	0,26.6417	30000	1.3e-15	2.2e-16

ERROR MESSAGES:

message	condition	value returned
underflow	x > 9.231948545 (DEC)	0.0

dawsn

Dawson's Integral.

SYNOPSIS:

```
double x, y, dawsn();
y = dawsn(x);
```

DESCRIPTION:

Approximates the integral

$$\text{dawsn}(x) = \exp(-x^2) \int_0^x \exp(t^2)\, dt$$

Three different rational approximations are employed, for the intervals 0 to 3.25; 3.25 to 6.25; and 6.25 up.

ACCURACY:

			Relative error:	
arithmetic	domain	# trials	peak	rms
IEEE	0,10	10000	6.9e-16	1.0e-16
DEC	0,10	6000	7.4e-17	1.4e-17

fresnl

Fresnel integral.

SYNOPSIS:

```
double x, S, C;
void fresnl();
fresnl(x, _&S, _&C);
```

DESCRIPTION:

Evaluates the Fresnel integrals

```
          x
          -
         | |
C(x) =   |   cos(pi/2 t**2) dt,
       | |
       -
        0


          x
          -
         | |
S(x) =   |   sin(pi/2 t**2) dt.
       | |
       -
        0
```

The integrals are evaluated by a power series for x < 1. For x >= 1 auxiliary functions f(x) and g(x) are employed such that:

```
C(x) = 0.5 + f(x) sin(pi/2 x**2) - g(x) cos(pi/2 x**2)
S(x) = 0.5 - f(x) cos(pi/2 x**2) - g(x) sin(pi/2 x**2)
```

ACCURACY:

Relative error.

Arithmetic	function	domain	# trials	peak	rms
IEEE	S(x)	0, 10	10000	2.0e-15	3.2e-16
IEEE	C(x)	0, 10	10000	1.8e-15	3.3e-16
DEC	S(x)	0, 10	6000	2.2e-16	3.9e-17
DEC	C(x)	0, 10	5000	2.3e-16	3.9e-17

JavaScript:

```
var x=  2.5625;
var r = cephes.fresnl(x);
Session,Output(r.result);
Session,Output(r.ssa);
Session,Output(r.csa);
```

Return value: Object
Format: JSON

```
{
  "result" : int,
  "ssa" : double,
  "cca" : double
}
```

贝塞尔

- airy - Airy函数
- j0 - 贝塞尔，0 阶
- j1 - 贝塞尔， 1阶
- jn - 贝塞尔，n 阶
- jv - 贝塞尔，非整数阶
- y0 - 贝塞尔，第二类，0 阶
- y1 - 贝塞尔，第二类， 1阶
- yn - 贝塞尔，第二类，n 阶
- yv - 贝塞尔，非整数阶
- i0 - 修正贝塞尔，0 阶
- i0e - 指数缩放的i0
- i1 - 修正贝塞尔， 1阶
- i1e - 以指数方式缩放的i1
- iv - 修正贝塞尔，非整数。命令
- k0 - 修正贝塞尔，第 3 类，0 阶
- k0e - 指数缩放k0
- k1 - 修正贝塞尔，第 3 类， 1阶
- k1e - 指数缩放的k1
- kn - 修正贝塞尔，第 3 类，阶数 n

airy

Airy function.

SYNOPSIS:

```
double x, ai, aip, bi, bip;
int airy();
airy(x, _&ai, _&aip, _&bi, _&bip);
```

DESCRIPTION:

Solution of the differential equation:

y"(x) = xy.

The function returns the two independent solutions Ai, Bi and their first derivatives Ai'(x), Bi'(x).
Evaluation is by power series summation for small x, by rational minimax approximations for large x.

ACCURACY:

Error criterion is absolute when function <= 1, relative when function > 1, except * denotes relative error criterion.
For large negative x, the absolute error increases as x^1.5.
For large positive x, the relative error increases as x^1.5.

Arithmetic	domain	function	# trials	peak	rms
IEEE	-10, 0	Ai	10000	1.6e-15	2.7e-16
IEEE	0, 10	Ai	10000	2.3e-14*	1.8e-15*
IEEE	-10, 0	Ai'	10000	4.6e-15	7.6e-16
IEEE	0, 10	Ai'	10000	1.8e-14*	1.5e-15*
IEEE	-10, 10	Bi	30000	4.2e-15	5.3e-16
IEEE	-10, 10	Bi'	30000	4.9e-15	7.3e-16
DEC	-10, 0	Ai	5000	1.7e-16	2.8e-17
DEC	0, 10	Ai	5000	2.1e-15*	1.7e-16*
DEC	-10, 0	Ai'	5000	4.7e-16	7.8e-17
DEC	0, 10	Ai'	12000	1.8e-15*	1.5e-16*
DEC	-10, 10	Bi	10000	5.5e-16	6.8e-17
DEC	-10, 10	Bi'	7000	5.3e-16	8.7e-17

JavaScript:

```
var x = 9.50313909;
var a = cephes.airy(x);
```

Return value: Object

Format: JSON

```
{
  "result" : integer,
  "ai"  : double,
  "aip" : double.
  "bi"  : double,
  "bip" : double
}
```

j0

Bessel function of order zero.

SYNOPSIS:

double x, y, j0();
y = j0(x);

DESCRIPTION:

Returns a Bessel function of order zero of the argument. The domain is divided into the intervals [0, 5] and (5, infinity). In the first interval this rational approximation is used:

$$(w - r_1^2)(w - r_2^2) P(w) / Q(w)$$

where $w = x^2$ and each r is a zero of the function.

In the second interval, the Hankel asymptotic expansion is employed with two rational functions of degree 6/6 and 7/7.

ACCURACY:

		Absolute error:		
arithmetic	domain	# trials	peak	rms
DEC	0, 30	10000	4.4e-17	6.3e-18
IEEE	0, 30	60000	4.2e-16	1.1e-16

j1

Bessel function of order one.

SYNOPSIS:

double x, y, j1();
y = j1(x);

DESCRIPTION:

Returns a Bessel function of order one of the argument.

The domain is divided into the intervals [0, 8] and (8, infinity). In the first interval a 24 term Chebyshev expansion is used. In the second, the asymptotic trigonometric representation is employed, using two rational functions of degree 5/5.

ACCURACY:

		Absolute error:		
arithmetic	domain	# trials	peak	rms
DEC	0, 30	10000	4.0e-17	1.1e-17
IEEE	0, 30	30000	2.6e-16	1.1e-16

jn

Bessel function of integer order.

SYNOPSIS:

int n;
double x, y, jn();
y = jn(n, x);

DESCRIPTION:

Returns a Bessel function of order n, where n is a (possibly negative) integer.

The ratio of jn(x) to j0(x) is computed by backward recurrence. First the ratio jn/jn-1 is found by a continued fraction expansion. Then the recurrence relating successive orders is applied until j0 or j1 is reached.

If n = 0 or 1 the routine for j0 or j1 is called directly.

ACCURACY:

Absolute error:

arithmetic	range	# trials	peak	rms
DEC	0, 30	5500	6.9e-17	9.3e-18
IEEE	0, 30	5000	4.4e-16	7.9e-17

Not suitable for large n or x. Use jv() instead.

jv

Bessel function of non-integer order.

SYNOPSIS:

double v, x, y, jv();
y = jv(v, x);

DESCRIPTION:

Returns a Bessel function of order v of the argument, where v is real. Negative x is allowed if v is an integer.

Several expansions are included: the ascending power series, the Hankel expansion, and two transitional expansions for large v. If v is not too large, it is reduced by recurrence to a region of best accuracy. The transitional expansions give 12D accuracy for v > 500.

ACCURACY:

Results for integer v are indicated by *, where x and v both vary from -125 to +125. Otherwise, x ranges from 0 to 125, v ranges as indicated by "domain." Error criterion is absolute, except relative when |jv()| > 1.

arithmetic	v domain	x domain	# trials	peak	rms
IEEE	0,125	0,125	100000	4.6e-15	2.2e-16
IEEE	-125,0	0,125	40000	5.4e-11	3.7e-13
IEEE	0,500	0,500	20000	4.4e-15	4.0e-16
Integer v:					
IEEE	-125,125	-125,125	50000	3.5e-15*	1.9e-16*

y0

Bessel function of the second kind, order zero, of the argument.

SYNOPSIS:

double x, y, y0();
y = y0(x);

DESCRIPTION:

Returns a Bessel function of the second kind, of order zero, of the argument.
The domain is divided into the intervals [0, 5] and (5, infinity). In the first interval a rational approximation R(x) is employed to compute:

y0(x) = R(x) + 2 * log(x) * j0(x) / PI.

Thus a call to j0() is required.

In the second interval, the Hankel asymptotic expansion is employed with two rational functions of degree 6/6 and 7/7.

ACCURACY:

Absolute error, when y0(x) < 1; else relative error:

arithmetic	domain	# trials	peak	rms
DEC	0, 30	9400	7.0e-17	7.9e-18
IEEE	0, 30	30000	1.3e-15	1.6e-16

y1

Bessel function of second kind of order one.

SYNOPSIS:

double x, y, y1();
y = y1(x);

DESCRIPTION:

Returns a Bessel function of the second kind of order one of the argument.

The domain is divided into the intervals [0, 8] and (8, infinity). In the first interval a 25 term Chebyshev expansion is used, and a call to j1() is required. In the second, the asymptotic trigonometric representation is employed using two rational functions of degree 5/5.

ACCURACY:

	Absolute error:			
arithmetic	domain	# trials	peak	rms
DEC	0, 30	10000	8.6e-17	1.3e-17
IEEE	0, 30	30000	1.0e-15	1.3e-16

(error criterion relative when |y1| > 1).

yn

Bessel function of second kind of integer order.

SYNOPSIS:

```
double x, y, yn();
int n;
y = yn(n, x);
```

DESCRIPTION:

Returns a Bessel function of order n, where n is a (possibly negative) integer.
The function is evaluated by forward recurrence on n, starting with values computed by the routines y0() and y1().
If n = 0 or 1 the routine for y0 or y1 is called directly.

ACCURACY:

	Absolute error, except relative when y > 1:			
arithmetic	domain	# trials	peak	rms
DEC	0, 30	2200	2.9e-16	5.3e-17
IEEE	0, 30	30000	3.4e-15	4.3e-16

ERROR MESSAGES:

message	condition	value returned
singularity	x = 0	MAXNUM
overflow		MAXNUM

Spot checked against tables for x, n between 0 and 100.

yv

Bessel function of the second kind, of non-integer order.

SYNOPSIS:

double v, x, y, yv();
y = yv(v, x);

DESCRIPTION:

Returns a Bessel function of the second kind, of order v of the argument, where v is a non-integer.

ACCURACY:

Not accurately characterized, but spot checked against tables.

i0

Modified Bessel function of order zero.

SYNOPSIS:

```
double x, y, i0();
y = i0(x);
```

DESCRIPTION:

Returns a modified Bessel function of order zero of the argument.

The function is defined as i0(x) = j0(ix).

The range is partitioned into the two intervals [0,8] and (8, infinity). Chebyshev polynomial expansions are employed in each interval.

ACCURACY:

		Relative error:		
arithmetic	domain	# trials	peak	rms
DEC	0,30	6000	8.2e-17	1.9e-17
IEEE	0,30	30000	5.8e-16	1.4e-16

i0e

Modified Bessel function of order zero, exponentially scaled.

SYNOPSIS:

```
double x, y, i0e();
y = i0e(x);
```

DESCRIPTION:

Returns exponentially scaled modified Bessel function of order zero of the argument.
The function is defined as i0e(x) = exp(-|x|) j0(ix).

ACCURACY:

		Relative error:		
arithmetic	domain	# trials	peak	rms
IEEE	0,30	30000	5.4e-16	1.2e-16

i1

Modified Bessel function of order one.

SYNOPSIS:

double x, y, i1();
y = i1(x);

DESCRIPTION:

Returns the modified Bessel function of order one of the argument.

The function is defined as i1(x) = -i j1(ix).

The range is partitioned into the two intervals [0,8] and (8, infinity). Chebyshev polynomial expansions are employed in each interval.

ACCURACY:

	Relative error:			
arithmetic	domain	# trials	peak	rms
DEC	0, 30	3400	1.2e-16	2.3e-17
IEEE	0, 30	30000	1.9e-15	2.1e-16

i1e

Modified Bessel function of order one, exponentially scaled.

SYNOPSIS:

```
double x, y, i1e();
y = i1e(x);
```

DESCRIPTION:

Returns the exponentially scaled modified Bessel function of order one of the argument.
The function is defined as i1(x) = -i exp(-|x|) j1(ix).

ACCURACY:

		Relative error:		
arithmetic	domain	# trials	peak	rms
IEEE	0, 30	30000	2.0e-15	2.0e-16

iv

Modified Bessel function of noninteger order.

SYNOPSIS:

double v, x, y, iv();
y = iv(v, x);

DESCRIPTION:

Returns the modified Bessel function of order v of the argument. If x is negative, v must be integer-valued.

The function is defined as Iv(x) = Jv(ix). Here, it is computed in terms of the confluent hypergeometric function, according to the formula:

$$Iv(x) = (x/2)^v e^{-x} \, hyperg(v+0.5, 2v+1, 2x) / gamma(v+1)$$

If v is a negative integer, then v is replaced by -v.

ACCURACY:

Tested at random points (v, x), with v between 0 and 30, x between 0 and 28.

		Relative error:		
arithmetic	domain	# trials	peak	rms
DEC	0,30	2000	3.1e-15	5.4e-16
IEEE	0,30	10000	1.7e-14	2.7e-15

Accuracy is diminished if v is near a negative integer.

k0

Modified Bessel function, third kind, order zero.

SYNOPSIS:

double x, y, k0();
y = k0(x);

DESCRIPTION:

Returns the modified Bessel function of the third kind of order zero of the argument.

The range is partitioned into the two intervals [0,8] and (8, infinity). Chebyshev polynomial expansions are employed in each interval.

ACCURACY:

Tested at 2000 random points between 0 and 8. Peak absolute error (relative when K0 > 1) was 1.46e-14; rms, 4.26e-15.

	Relative error:			
arithmetic	domain	# trials	peak	rms
DEC	0, 30	3100	1.3e-16	2.1e-17
IEEE	0, 30	30000	1.2e-15	1.6e-16

ERROR MESSAGES:

message	condition	value returned
domain	x <= 0	MAXNUM

k0e

Modified Bessel function, third kind, order zero, exponentially scaled.

SYNOPSIS:

```
double x, y, k0e();
y = k0e(x);
```

DESCRIPTION:

Returns the exponentially scaled, modified Bessel function of the third kind of order zero of the argument.

ACCURACY:

		Relative error:		
arithmetic	domain	# trials	peak	rms
IEEE	0, 30	30000	1.4e-15	1.4e-16

k1

Modified Bessel function, third kind, order one.

SYNOPSIS:

double x, y, k1();
y = k1(x);

DESCRIPTION:

Computes the modified Bessel function of the third kind, of order one of the argument.

The range is partitioned into the two intervals [0,2] and (2, infinity). Chebyshev polynomial expansions are employed in each interval.

ACCURACY:

		Relative error:		
arithmetic	domain	# trials	peak	rms
DEC	0, 30	3300	8.9e-17	2.2e-17
IEEE	0, 30	30000	1.2e-15	1.6e-16

ERROR MESSAGES:

message	condition	value returned
domain	x <= 0	MAXNUM

k1e

Modified Bessel function, third kind, order one, exponentially scaled.

SYNOPSIS:

```
double x, y, k1e();
y = k1e(x);
```

DESCRIPTION:

Returns the exponentially scaled, modified Bessel function of the third kind of order one of the argument:

```
k1e(x) = exp(x) * k1(x).
```

ACCURACY:

		Relative error:		
arithmetic	domain	# trials	peak	rms
IEEE	0, 30	30000	7.8e-16	1.2e-16

kn

Modified Bessel function, third kind, integer order.

SYNOPSIS:

double x, y, kn();
int n;
y = kn(n, x);

DESCRIPTION:

Returns the modified Bessel function of the third kind, of order n of the argument.

The range is partitioned into the two intervals [0,9.55] and (9.55, infinity). An ascending power series is used in the low range, and an asymptotic expansion in the high range.

ACCURACY:

		Relative error:		
arithmetic	domain	# trials	peak	rms
DEC	0,30	3000	1.3e-9	5.8e-11
IEEE	0,30	90000	1.8e-8	3.0e-10

Error is high only near the crossover point x = 9.55 between the two expansions used.

超几何

- hyperg - 汇合超几何
- hyp2f1 - 高斯超几何函数
- hyp2f0 - 2F0
- onef2 - 1F2
- 三f0 - threef0

hyperg

Confluent hypergeometric function.

SYNOPSIS:

double a, b, x, y, hyperg();
y = hyperg(a, b, x);

DESCRIPTION:

Computes the confluent hypergeometric function

$$F(a,b;x) = 1 + \frac{a\,x}{b\,1!} + \frac{a(a+1)\,x^2}{b(b+1)\,2!} + \ldots$$

Many higher transcendental functions are special cases of this power series.

As is evident from the formula, b must not be a negative integer or zero unless a is an integer with 0 >= a > b.

The routine attempts both a direct summation of the series and an asymptotic expansion. In each case error due to roundoff, cancellation and nonconvergence is estimated. The result with smaller estimated error is returned.

ACCURACY:

Tested at random points (a, b, x), all three variables ranging from 0 to 30.

			Relative error:	
arithmetic	domain	# trials	peak	rms
DEC	0,30	2000	1.2e-15	1.3e-16

qtst1:

21800 max = 1.4200E-14 rms = 1.0841E-15 ave = -5.3640E-17

ltstd:

25500 max = 1.2759e-14 rms = 3.7155e-16 ave = 1.5384e-18

IEEE	0,30	30000	1.8e-14	1.1e-15

Larger errors can be observed when b is near a negative integer or zero. Certain combinations of arguments yield serious cancellation errors in the power series summation and also are not in the region of near convergence of the asymptotic series. An error message is printed if the self-estimated relative error is greater than 1.0e-12.

hyp2f1

Gauss hypergeometric function 2F1.

SYNOPSIS:

```
double a, b, c, x, y, hyp2f1();
y = hyp2f1(a, b, c, x);
```

DESCRIPTION:

```
 hyp2f1( a, b, c, x )  =   F ( a, b; c; x )
                         2 1

          inf.
           -   a(a+1)...(a+k) b(b+1)...(b+k)   k+1
  =  1 +   >   ----------------------------  x   .
           -         c(c+1)...(c+k) (k+1)!
         k = 0
```

Cases addressed are:

- Tests and escapes for negative integer a, b, or c
- Linear transformation if c - a or c - b negative integer
- Special case c = a or c = b
- Linear transformation for x near +1
- Transformation for x < -0.5
- Psi function expansion if x > 0.5 and c - a - b integer Conditionally, a recurrence on c to make c-a-b > 0

|x| > 1 is rejected.

The parameters a, b, c are considered to be integer valued if they are within 1.0e-14 of the nearest integer (1.0e-13 for IEEE arithmetic).

ACCURACY:

Relative error (-1 < x < 1):

arithmetic	domain	# trials	peak	rms
IEEE	-1,7	230000	1.2e-11	5.2e-14

Several special cases also tested with a, b, c in the range -7 to 7.

ERROR MESSAGES:

A "partial loss of precision" message is printed if the internally estimated relative error exceeds 1^-12.

A "singularity" message is printed on overflow or in cases not addressed (such as x < -1).

hyp2f0

See the hyperg Help topic.

onef2

请参阅struve帮助。

threef0

请参阅struve帮助。

椭圆

- ellpe -完成椭圆积分 (E)
- ellie - 不完全椭圆积分 (E)
- ellpk -完成椭圆积分 （ K ）
- ellik - 不完全椭圆积分 (K)
- ellpj - 雅可比椭圆函数

ellpe

Complete elliptic integral of the second kind.

SYNOPSIS:

```
double m1, y, ellpe();
y = ellpe(m1);
```

DESCRIPTION:

Approximates the integral

```
           pi/2
            -
           | |              2
E(m)  =    |    sqrt( 1 - m sin t ) dt
         | |
          -
           0
```

Where m = 1 - m1, using the approximation:

P(x) - x log x Q(x).

Though there are no singularities, the argument m1 is used rather than m, for compatibility with ellpk().

E(1) = 1; E(0) = pi/2.

ACCURACY:

		Relative error:		
arithmetic	domain	# trials	peak	rms
DEC	0, 1	13000	3.1e-17	9.4e-18
IEEE	0, 1	10000	2.1e-16	7.3e-17

ERROR MESSAGES:

message	condition	value returned
domain	x<0, x>1	0.0

ellie

Incomplete elliptic integral of the second kind.

SYNOPSIS:

```
double phi, m, y, ellie();
y = ellie(phi, m);
```

DESCRIPTION:

Approximates the integral:

```
                phi
                 -
                | |
                |                2
E(phi_\m)  =    |    sqrt(1 - m sin t) dt
                |
              | |
              -
               0
```

of amplitude phi and modulus m, using the arithmetic - geometric mean algorithm.

ACCURACY:

Tested at random arguments with phi in [-10, 10] and m in [0, 1].

		Relative error:		
arithmetic	domain	# trials	peak	rms
DEC	0,2	2000	1.9e-16	3.4e-17
IEEE	-10,10	150000	3.3e-15	1.4e-16

ellpk

Complete elliptic integral of the first kind.

SYNOPSIS:

```
double m1, y, ellpk();
y = ellpk(m1);
```

DESCRIPTION:

Approximates the integral:

```
          pi/2
           -
          | |
          |          dt
K(m)  =   |    ------------------
          |                2
        | |    sqrt(1 - m sin t)
          -
          0
```

where m = 1 - m1, using the approximation:

P(x) - log x Q(x).

The argument m1 is used rather than m, so that the logarithmic singularity at m = 1 will be shifted to the origin; this preserves maximum accuracy.

K(0) = pi/2.

ACCURACY:

		Relative error:		
arithmetic	domain	# trials	peak	rms
DEC	0,1	16000	3.5e-17	1.1e-17
IEEE	0,1	30000	2.5e-16	6.8e-17

ERROR MESSAGES:

message	condition	value returned

domain	x<0, x>1	0.0

ellik

Incomplete elliptic integral of the first kind.

SYNOPSIS:

```
double phi, m, y, ellik();
y = ellik(phi, m);
```

DESCRIPTION:

Approximates the integral:

```
                phi
                 -
                | |
                |          dt
F(phi_\m)  =    |    ------------------
                |                  2
               | |    sqrt(1 - m sin t)
                -
                0
```

of amplitude phi and modulus m, using the arithmetic - geometric mean algorithm.

ACCURACY:

Tested at random points with m in [0, 1] and phi as indicated.

```
                    Relative error:
arithmetic   domain     # trials      peak         rms
```

ellpj

Jacobian Elliptic Functions.

SYNOPSIS:

double u, m, sn, cn, dn, phi;

int ellpj();

ellpj(u, m, _&sn, _&cn, _&dn, _&phi);

DESCRIPTION:

Evaluates the Jacobian elliptic functions sn(u|m), cn(u|m), and dn(u|m) of parameter m between 0 and 1, and real argument u.

These functions are periodic, with quarter-period on the real axis equal to the complete elliptic integral ellpk(1.0-m).

Relation to incomplete elliptic integral:

If u = ellik(phi,m), then sn(u|m) = sin(phi), and cn(u|m) = cos(phi).

Phi is called the amplitude of u.

Computation is by means of the arithmetic-geometric mean algorithm, except when m is within 1e-9 of 0 or 1.

In the latter case with m close to 1, the approximation applies only for phi < pi/2.

ACCURACY:

Tested at random points with u between 0 and 10, m between 0 and 1.

Absolute error (* = relative error):

arithmetic	function	# trials	peak	rms
DEC	sn	1800	4.5e-16	8.7e-17
IEEE	phi	10000	9.2e-16*	1.4e-16*
IEEE	sn	50000	4.1e-15	4.6e-16
IEEE	cn	40000	3.6e-15	4.4e-16
IEEE	dn	100000	3.9e-15	1.7e-16

Larger errors occur for m near 1.

Peak error observed in consistency check using addition theorem for sn(u+v) was 4e-16 (absolute). Also tested by the earlier relation to the incomplete elliptic integral. Accuracy deteriorates when u is large.

概率

- bdtr - 二项分布
- bdtrc - 补充二项式
- bdtri - 逆二项式
- chdtr - 卡方分布
- chdtrc - 补充卡方
- chdtri - 反卡方
- fdtr - F 分布
- fdtrc - 补充 F
- fdtri - 逆 F 分布
- gdtr - Gamma分布
- gdtrc - 补充gamma
- nbdtr - 负二项分布
- nbdtrc - 补负二项式
- ndtr - 正态分布
- ndtri - 逆正态分布
- pdtr - 泊松分布
- pdtrc - 补泊松
- pdtri - 逆泊松分布
- stdtr - 学生的 t 分布

bdtr

Binomial distribution.

SYNOPSIS:

```
int k, n;
double p, y, bdtr();
y = bdtr(k, n, p);
```

DESCRIPTION:

Returns the sum of the terms 0 through k of the Binomial probability density:

```
  k
  --  ( n )   j      n-j
  >   (   )  p  (1-p)
  --  ( j )
 j=0
```

The terms are not summed directly; instead the incomplete beta integral is employed, according to the formula:

y = bdtr(k, n, p) = incbet(n-k, k+1, 1-p).

The arguments must be positive, with p ranging from 0 to 1.

ACCURACY:

Tested at random points (a,b,p), with p between 0 and 1.

	a,b		Relative error:	
arithmetic	domain	# trials	peak	rms
For p between 0.001 and 1:				
IEEE	0,100	100000	4.3e-15	2.6e-16

ERROR MESSAGES:

message	condition	value returned
domain	k < 0	0.0
	n < k	
	x < 0, x > 1	

bdtrc

Complemented binomial distribution.

SYNOPSIS:

```
int k, n;
double p, y, bdtrc();
y = bdtrc(k, n, p);
```

DESCRIPTION:

Returns the sum of the terms k+1 through n of the Binomial
probability density:

```
  n
  --  ( n )   j      n-j
  >   (   )  p  (1-p)
  --  ( j )
 j=k+1
```

The terms are not summed directly; instead the incomplete beta integral is employed, according to the formula:

y = bdtrc(k, n, p) = incbet(k+1, n-k, p).

The arguments must be positive, with p ranging from 0 to 1.

ACCURACY:

Tested at random points (a,b,p).

```
              a,b                     Relative error:
arithmetic  domain     # trials      peak         rms
  For p between 0.001 and 1:
     IEEE     0,100       100000      6.7e-15     8.2e-16
  For p between 0 and .001:
     IEEE     0,100       100000      1.5e-13     2.7e-15
```

ERROR MESSAGES:

```
  message         condition      value returned
```

domain	x<0, x>1, n<k	0.0

bdtri

Inverse binomial distribution.

SYNOPSIS:

int k, n;
double p, y, bdtri();
p = bdtr(k, n, y);

DESCRIPTION:

Finds the event probability p such that the sum of the terms 0 through k of the Binomial probability density is equal to the given cumulative probability y.
This is accomplished using the inverse beta integral function and the relation:

1 - p = incbi(n-k, k+1, y).

ACCURACY:

Tested at random points (a,b,p).

	a,b		Relative error:	
arithmetic	domain	# trials	peak	rms
For p between 0.001 and 1:				
IEEE	0,100	100000	2.3e-14	6.4e-16
IEEE	0,10000	100000	6.6e-12	1.2e-13
For p between 10^-6 and 0.001:				
IEEE	0,100	100000	2.0e-12	1.3e-14
IEEE	0,10000	100000	1.5e-12	3.2e-14

See the incbi Help topic.

ERROR MESSAGES:

message	condition	value returned
domain	k < 0, n <= k	0.0
	x < 0, x > 1	

chdtr

Chi-square distribution.

SYNOPSIS:

double df, x, y, chdtr();
y = chdtr(df, x);

DESCRIPTION:

Returns the area under the left hand tail (from 0 to x) of the Chi square probability density function, with v degrees of freedom.

$$P(x \mid v) = \frac{1}{2^{v/2}\,\Gamma(v/2)} \int_0^x t^{v/2-1} e^{-t/2}\, dt$$

where x is the Chi-square variable.

The incomplete gamma integral is used, according to the formula:

y = chdtr(v, x) = igam(v/2.0, x/2.0).

The arguments must both be positive.

ACCURACY:

See the igam() Help topic.

ERROR MESSAGES:

message	condition	value returned
domain	x < 0 or v < 1	0.0

chdtrc

Complemented Chi-square distribution.

SYNOPSIS:

double v, x, y, chdtrc();
y = chdtrc(v, x);

DESCRIPTION:

Returns the area under the right hand tail (from x to infinity) of the Chi square probability density function with v degrees of freedom:

$$P(x \mid v) = \frac{1}{2^{v/2}\,\Gamma(v/2)} \int_{x}^{\infty} t^{v/2-1} e^{-t/2}\, dt$$

where x is the Chi-square variable.

The incomplete gamma integral is used, according to the formula:

y = chdtr(v, x) = igamc(v/2.0, x/2.0).

The arguments must both be positive.

ACCURACY:

See the igamc() Help topic.

ERROR MESSAGES:

message	condition	value returned
domain	x < 0 or v < 1	0.0

chdtri

Inverse of complemented Chi-square distribution.

SYNOPSIS:

double df, x, y, chdtri();
x = chdtri(df, y);

DESCRIPTION:

Finds the Chi-square argument x, such that the integral from x to infinity of the Chi-square density is equal to the given cumulative probability y.

This is accomplished using the inverse gamma integral function and the relation:

x/2 = igami(df/2, y);

ACCURACY:

See the igami Help topic.

ERROR MESSAGES:

message	condition	value returned
domain	y < 0 or y > 1	0.0
	v < 1	

fdtr

F distribution.

SYNOPSIS:

int df1, df2;
double x, y, fdtr();
y = fdtr(df1, df2, x);

DESCRIPTION:

Returns the area from zero to x under the F density function (also known as Snedcor's density, or the variance ratio density).
This is the density of x = (u1/df1)/(u2/df2), where u1 and u2 are random variables having Chi square distributions with df1 and df2 degrees of freedom, respectively.

The incomplete beta integral is used, according to the formula

P(x) = incbet(df1/2, df2/2, (df1*x/(df2 + df1*x)).

The arguments a and b are greater than zero, and x is nonnegative.

ACCURACY:

Tested at random points (a,b,x).

	x	a,b	Relative error:		
arithmetic	domain	domain	# trials	peak	rms
IEEE	0,1	0,100	100000	9.8e-15	1.7e-15
IEEE	1,5	0,100	100000	6.5e-15	3.5e-16
IEEE	0,1	1,10000	100000	2.2e-11	3.3e-12
IEEE	1,5	1,10000	100000	1.1e-11	1.7e-13

See the incbet Help topic.

ERROR MESSAGES:

message	condition	value returned
domain	a<0, b<0, x<0	0.0

fdtrc

Complemented F distribution.

SYNOPSIS:

```
int df1, df2;
double x, y, fdtrc();
y = fdtrc(df1, df2, x);
```

DESCRIPTION:

Returns the area from x to infinity under the F density function (also known as Snedcor's density or the variance ratio density).

```
                 inf.
                  -
          1      | |  a-1      b-1
1-P(x)  =  ------   |   t    (1-t)    dt
        B(a,b)  | |
                -
                 x
```

The incomplete beta integral is used, according to the formula

```
        P(x) = incbet( df2/2, df1/2, (df2/(df2 + df1*x) ).
```

ACCURACY:

Tested at random points (a,b,x) in the indicated intervals.

	x	a,b		Relative error:	
arithmetic	domain	domain	# trials	peak	rms
IEEE	0,1	1,100	100000	3.7e-14	5.9e-16
IEEE	1,5	1,100	100000	8.0e-15	1.6e-15
IEEE	0,1	1,10000	100000	1.8e-11	3.5e-13
IEEE	1,5	1,10000	100000	2.0e-11	3.0e-12

ERROR MESSAGES:

message	condition	value returned
domain	a<0, b<0, x<0	0.0

fdtri

Inverse of complemented F distribution.

SYNOPSIS:

```
int df1, df2;
double x, p, fdtri();
x = fdtri(df1, df2, p);
```

DESCRIPTION:

Finds the F density argument x, such that the integral from x to infinity of the F density is equal to the given probability p.
This is accomplished using the inverse beta integral function and the relations:

```
z = incbi(df2/2, df1/2, p)
x = df2 (1-z) / (df1 z).
```

Note: These relations hold for the inverse of the uncomplemented F distribution:

```
z = incbi(df1/2, df2/2, p)
x = df2 z / (df1 (1-z)).
```

ACCURACY:

Tested at random points (a,b,p).

	a,b		Relative error:	
arithmetic	domain	# trials	peak	rms
For p between .001 and 1:				
IEEE	1,100	100000	8.3e-15	4.7e-16
IEEE	1,10000	100000	2.1e-11	1.4e-13
For p between 10^-6 and 10^-3:				
IEEE	1,100	50000	1.3e-12	8.4e-15
IEEE	1,10000	50000	3.0e-12	4.8e-14

See the fdtrc Help topic.

ERROR MESSAGES:

message	condition	value returned

domain	p <= 0 or p > 1 v < 1	0.0

gdtr

Gamma distribution function.

SYNOPSIS:

double a, b, x, y, gdtr();
y = gdtr(a, b, x);

DESCRIPTION:

Returns the integral from zero to x of the gamma probability density function:

```
              x
       b      -
      a       | |   b-1  -at
y =  -----    |    t    e    dt
      -     | |
     | (b)   -
             0
```

The incomplete gamma integral is used, according to the relation:

y = igam(b, ax).

ACCURACY:

See the igam() Help topic.

ERROR MESSAGES:

message	condition	value returned
domain	x < 0	0.0

gdtrc

Complemented gamma distribution function.

SYNOPSIS:

double a, b, x, y, gdtrc();
y = gdtrc(a, b, x);

DESCRIPTION:

Returns the integral from x to infinity of the gamma probability density function:

```
              inf.
       b       -
      a       | |   b-1  -at
y =  -----    |    t    e    dt
      -     | |
     | (b)   -
              x
```

The incomplete gamma integral is used, according to the relation:

y = igamc(b, ax).

ACCURACY:

See the igamc() Help topic.

ERROR MESSAGES:

message	condition	value returned
domain	x < 0	0.0

nbdtr

Negative binomial distribution.

SYNOPSIS:

```
int k, n;
double p, y, nbdtr();
y = nbdtr(k, n, p);
```

DESCRIPTION:

Returns the sum of the terms 0 through k of the negative binomial distribution:

$$\sum_{j=0}^{k} \binom{n+j-1}{j} p^n (1-p)^j$$

In a sequence of Bernoulli trials, this is the probability that k or fewer failures precede the nth success.

The terms are not computed individually; instead the incomplete beta integral is employed, according to the formula:

y = nbdtr(k, n, p) = incbet(n, k+1, p).

The arguments must be positive, with p ranging from 0 to 1.

ACCURACY:

Tested at random points (a,b,p), with p between 0 and 1.

	a,b		Relative error:	
arithmetic	domain	# trials	peak	rms
IEEE	0,100	100000	1.7e-13	8.8e-15

See the incbet Help topic.

nbdtrc

Complemented negative binomial distribution.

SYNOPSIS:

```
int k, n;
double p, y, nbdtrc();
y = nbdtrc(k, n, p);
```

DESCRIPTION:

Returns the sum of the terms k+1 to infinity of the negative binomial distribution:

```
  inf
  --  (n+j-1)   n      j
  >   (     )  p  (1-p)
  --  (  j  )
 j=k+1
```

The terms are not computed individually; instead the incomplete beta integral is employed, according to the formula:

y = nbdtrc(k, n, p) = incbet(k+1, n, 1-p).

The arguments must be positive, with p ranging from 0 to 1.

ACCURACY:

Tested at random points (a,b,p), with p between 0 and 1.

	a,b		Relative error:	
arithmetic	domain	# trials	peak	rms
IEEE	0,100	100000	1.7e-13	8.8e-15

See the incbet Help topic.

ndtr

Normal distribution function.

SYNOPSIS:

```
double x, y, ndtr();
y = ndtr(x);
```

DESCRIPTION:

Returns the area under the Gaussian probability density function, integrated from minus infinity to x:

```
                           x
                          -
               1          | |          2
   ndtr(x)  = ---------    |    exp(- t /2) dt
              sqrt(2pi)  | |
                         -
                        -inf.

            =  (1 + erf(z)) / 2
            =  erfc(z) / 2
```

where z = x/sqrt(2).

Computation is via the functions erf and erfc, with care to avoid error amplification in computing exp(-x^2).

ACCURACY:

		Relative error:		
arithmetic	domain	# trials	peak	rms
IEEE	-13,0	30000	1.3e-15	2.2e-16

ERROR MESSAGES:

message	condition	value returned
underflow	x > 37.519379347	0.0

ndtri

Inverse of Normal distribution function.

SYNOPSIS:

double x, y, ndtri();
x = ndtri(y);

DESCRIPTION:

Returns the argument, x, for which the area under the Gaussian probability density function (integrated from minus infinity to x) is equal to y.

For small arguments 0 < y < exp(-2), the program computes z = sqrt(-2.0 * log(y)); then the approximation is x = z - log(z)/z - (1/z) P(1/z) / Q(1/z).

There are two rational functions P/Q, one for 0 < y < exp(-32) and the other for y up to exp(-2).

For larger arguments, w = y - 0.5, and x/sqrt(2pi) = w + w**3 R(w**2)/S(w**2)).

ACCURACY:

		Relative error:		
arithmetic	domain	# trials	peak	rms
DEC	0.125, 1	5500	9.5e-17	2.1e-17
DEC	6e-39, 0.135	3500	5.7e-17	1.3e-17
IEEE	0.125, 1	20000	7.2e-16	1.3e-16
IEEE	3e-308, 0.135	50000	4.6e-16	9.8e-17

ERROR MESSAGES:

message	condition	value returned
domain	x <= 0	-MAXNUM
domain	x >= 1	MAXNUM

pdtr

Poisson distribution.

SYNOPSIS:

```
int k;
double m, y, pdtr();
y = pdtr(k, m);
```

DESCRIPTION:

Returns the sum of the first k terms of the Poisson distribution:

$$\sum_{j=0}^{k} \frac{e^{-m} m^j}{j!}$$

The terms are not summed directly; instead the incomplete gamma integral is employed, according to the relation:

y = pdtr(k, m) = igamc(k+1, m).

The arguments must both be positive.

ACCURACY:
See the igamc Help topic.

pdtrc

Complemented poisson distribution.

SYNOPSIS:

```
int k;
double m, y, pdtrc();
y = pdtrc(k, m);
```

DESCRIPTION:

Returns the sum of the terms k+1 to infinity of the Poisson distribution:

$$\sum_{j=k+1}^{\infty} \frac{e^{-m} m^j}{j!}$$

The terms are not summed directly; instead the incomplete gamma integral is employed, according to the formula:

y = pdtrc(k, m) = igam(k+1, m).

The arguments must both be positive.

ACCURACY:
See the igam Help topic.

pdtri

Inverse Poisson distribution.

SYNOPSIS:

```
int k;
double m, y, pdtr();
m = pdtri(k, y);
```

DESCRIPTION:

Finds the Poisson variable x such that the integral from 0 to x of the Poisson density is equal to the given probability y. This is accomplished using the inverse gamma integral function and the relation

```
m = igami(k+1, y).
```

ACCURACY:

See the igami Help topic.

ERROR MESSAGES:

message	condition	value returned
domain	y < 0 or y >= 1	0.0
	k < 0	

stdtr

Student's t distribution.

SYNOPSIS:

```
double t, stdtr();
short k;
y = stdtr(k, t);
```

DESCRIPTION:

Computes the integral from minus infinity to t of the Student t distribution with integer k > 0 degrees of freedom:

$$\frac{\Gamma((k+1)/2)}{\sqrt{k\pi}\,\Gamma(k/2)} \int_{-\infty}^{t} \left(1 + \frac{x}{k}\right)^{-(k+1)/2} dx$$

Relation to incomplete beta integral:

$$1 - \mathrm{stdtr}(k,t) = 0.5 * \mathrm{incbet}(k/2, 1/2, z)$$

where

$$z = k/(k + t^{**}2).$$

For t < -2, this is the method of computation.

For higher t, a direct method is derived from integration by parts.

Since the function is symmetric about t=0, the area under the right tail of the density is found by calling the function with -t instead of t.

ACCURACY:

Tested at random 1 <= k <= 25. The 'domain' refers to t.

Relative error:

arithmetic	domain	# trials	peak	rms
IEEE	-100,-2	50000	5.9e-15	1.4e-15
IEEE	-2,100	500000	2.7e-15	4.9e-17

杂项

- polylog - 多对数
- spence - 对数
- zetac - 黎曼 Zeta函数
- zeta - 两个参数的zeta函数
- struve - Struve函数

polylog

Polylogarithms.

SYNOPSIS:

```
double x, y, polylog();
int n;
y = polylog(n, x);
```

The polylogarithm of order n is defined by the series:

$$\mathrm{Li}_n(x) = \sum_{k=1}^{\infty} \frac{x^k}{k^n}.$$

For x = 1,

$$\mathrm{Li}_n(1) = \sum_{k=1}^{\infty} \frac{1}{k^n} = \text{Riemann zeta function (n)}.$$

When n = 2, the function is the dilogarithm, related to Spence's integral:

$$\mathrm{Li}_2(x) = \int_0^x \frac{-\ln(1-t)}{t}\,dt = \int_1^{1-x} \frac{\ln t}{1-t}\,dt = \mathrm{spence}(1-x).$$

References:

Lewin, L., *Polylogarithms and Associated Functions*,
North Holland, 1981.

Lewin, L., ed., *Structural Properties of Polylogarithms*,
American Mathematical Society, 1991.

ACCURACY:

Relative error:

arithmetic	domain	n	# trials	peak	rms
IEEE	0, 1	2	50000	6.2e-16	8.0e-17
IEEE	0, 1	3	100000	2.5e-16	6.6e-17
IEEE	0, 1	4	30000	1.7e-16	4.9e-17
IEEE	0, 1	5	30000	5.1e-16	7.8e-17

spence

Dilogarithm.

SYNOPSIS:

```
double x, y, spence();
y = spence(x);
```

DESCRIPTION:

Computes the integral:

```
                  x
                  -
                 | | log t
spence(x)  =  -  |   ----- dt
              | |   t - 1
              -
              1
```

for x >= 0. A rational approximation gives the integral in the interval (0.5, 1.5). Transformation formulas for 1/x and 1-x are employed outside the basic expansion range.

ACCURACY:

		Relative error:		
arithmetic	domain	# trials	peak	rms
IEEE	0,4	30000	3.9e-15	5.4e-16
DEC	0,4	3000	2.5e-16	4.5e-17

zetac

Riemann zeta function.

SYNOPSIS:
double x, y, zetac();
y = zetac(x);

DESCRIPTION:

$$\text{zetac}(x) = \sum_{k=2}^{\infty} k^{-x}, \quad x > 1,$$

Is related to the Riemann zeta function by:

Riemann zeta(x) = zetac(x) + 1.

Extension of the function definition for x < 1 is implemented.
Zero is returned for x > log2(MAXNUM).

An overflow error might occur for large negative x, due to the gamma function in the reflection formula.

ACCURACY:
Tabulated values have full machine accuracy.

			Relative error:	
arithmetic	domain	# trials	peak	rms
IEEE	1,50	10000	9.8e-16	1.3e-16
DEC	1,50	2000	1.1e-16	1.9e-17

zeta

Riemann zeta function of two arguments.

SYNOPSIS:

```
double x, q, y, zeta();
y = zeta(x, q);
```

DESCRIPTION:

$$\zeta(x,q) = \sum_{k=0}^{\infty} (k+q)^{-x}$$

where x > 1 and q is not a negative integer or zero.
The Euler-Maclaurin summation formula is used to obtain the expansion

$$\zeta(x,q) = \sum_{k=1}^{n} (k+q)^{-x}$$

$$+ \frac{(n+q)^{1-x}}{x-1} + \frac{1}{2(n+q)^{x}} + \sum_{j=1}^{\infty} \frac{B_{2j}\, x(x+1)\cdots(x+2j)}{(2j)!\,(n+q)^{x+2j+1}}$$

where the B2j are Bernoulli numbers.
Note that zeta(x,1) = zetac(x) + 1.
(see zetac)

ACCURACY:

REFERENCE:

Gradshteyn, I. S., and I. M. Ryzhik, *Tables of Integrals, Series, and Products*, p. 1073; Academic Press, 1980.

struve

Struve function.

SYNOPSIS:

double v, x, y, struve();
y = struve(v, x);

DESCRIPTION:

Computes the Struve function Hv(x) of order v, argument x. Negative x is rejected unless v is an integer.

This module also contains the hypergeometric functions 1F2 and 3F0, and a routine for the Bessel function Yv(x) with noninteger v.

ACCURACY:

Not accurately characterized, but spot checked against tables.

矩阵

- fftr - 快速傅里叶变换
- simq - 联立线性方程组
- minv -矩阵反转
- mmmpy -矩阵乘法
- mvmpy -矩阵乘以向量
- mtransp -矩阵转置
- eigens - 特征向量（对称矩阵）

fftr

FFT of Real Valued Sequence.

SYNOPSIS:

double x[], sine[];

int m;

fftr(x, m, sine);

DESCRIPTION:

Computes the (complex valued) discrete Fourier transform of the real valued sequence x[]. The input sequence x[] contains n = 2**m samples. The program fills array sine[k] with n/4 + 1 values of sin(2 PI k / n).

Data format for complex valued output is real part followed by imaginary part. The output is developed in the input array x[].

The algorithm takes advantage of the fact that the FFT of an n point real sequence can be obtained from an n/2 point complex FFT.

A radix 2 FFT algorithm is used.

Execution time on an LSI-11/23 with floating point chip is 1.0 sec for n = 256.

REFERENCE:

E. Oran Brigham, *The Fast Fourier Transform*; Prentice-Hall, Inc., 1974

simq

Solution of simultaneous linear equations AX = B by Gaussian elimination with partial pivoting.

SYNOPSIS:

double A[n*n], B[n], X[n];

int n, flag;

int IPS[];

int simq();

ercode = simq(A, B, X, n, flag, IPS);

DESCRIPTION:

B, X, IPS are vectors of length n.

A is an n x n matrix (i.e. a vector of length n*n), stored row-wise; that is, A(i,j) = A[ij], where ij = i*n + j, which is the transpose of the normal column-wise storage.

The contents of matrix A are destroyed.

Set flag=0 to solve.

Set flag=-1 to do a new back substitution for a different B vector using the same A matrix previously reduced when flag=0.

The routine returns nonzero on error; messages are printed.

ACCURACY:

Depends on the conditioning (range of eigenvalues) of matrix A.

REFERENCE:

Computer Solution of Linear Algebraic Systems

by George E. Forsythe and Cleve B. Moler; Prentice-Hall, 1967.

minv

Matrix inversion.

SYNOPSIS:

```
int n, errcod;
double A[n*n], X[n*n];
double B[n];
int IPS[n];
int minv();

errcod = minv(A, X, n, B, IPS);
```

DESCRIPTION:

Finds the inverse of the n by n matrix A. The result goes to X. B and IPS are scratch-pad arrays of length n. The contents of matrix A are destroyed.

The routine returns nonzero on error; error messages are printed by the subroutine simq().

JavaScript:

```
function test_minv()
{
/*
* Finds the inverse of the n by n matrix A.  The result goes
* to X.   B and IPS are scratch pad arrays of length n.
* The contents of matrix A are destroyed
*/
	Session.Output("calling cephes.minv( A,X,n,B,IPS) where:");
	var n = 10; // n x n matrix A (10x10)
	var A = [
					[0.0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9],
					[0.0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9],
					[0.0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9],
					[0.0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9],
					[0.0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9],
					[0.0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9],
					[0.0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9],
					[0.0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9],
					[0.0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9],
```

```
                                [0.0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9]
        ];

        var X = new Array(10);     // output
        var B = new Array(10);      // scratch pad
        var IPS = new Array(10); // scratch pad

        Session.Output("  n = " + n);
        Session.Output("  length of A is" + n*n);
        Session.Output("A is matrix of " + dimensionsOfArray(A));
        var ir = cephes.minv(A,X,n,B,IPS);

        var s = cephes.geterrormsg();
        if(s.length>0)
        {
                Session.Output("error output by minv: " + s);
        }
        else
        {
                Session.Output("minv returned " + ir);
                Session.Output("X is matrix of " + dimensionsOfArray(X));
                printMatrix("X",X,10,10);
        }
}
```

mmmpy

Matrix-Matrix multiply

SYNOPSIS

```
int r, c;
double A[r*c], B[c*r], Y[r*r];
mmmpy( r, c, A, B, Y );
```

DESCRIPTION

Multiply an r (rows) by c (columns) matrix A on the left by a c (rows) by r (columns) matrix B on the right to produce an r by r matrix Y.

mvmpy

Matrix-Vector multiply

SYNOPSIS

```
int r, c;
double A[r*c], V[c], Y[r];
mvmpy( r, c, A, V, Y );
```

DESCRIPTION

Multiply r (rows) by c (columns) matrix A on the left by column vector V of dimension c on the right to produce a (column) vector Y output of dimension r.

mtransp

Matrix Transpose

SYNOPSIS

```
int n;
double A[n*n], T[n*n];
mtransp( n, A, T )
```

DESCRIPTION

Transpose the n by n square matrix A and put the result in T.
T can occupy the same storage as A.

eigens

Eigenvalues and eigenvectors of a real symmetric matrix.

SYNOPSIS:

int n;
double A[n*(n+1)/2], EV[n*n], E[n];
void eigens(A, EV, E, n);

DESCRIPTION:

The algorithm is due to J. vonNeumann.

- -

A[] is a symmetric matrix stored in lower triangular form. That is, A[row, column] = A[(row*row+row)/2 + column] or the equivalent with row and column interchanged. The indices row and column run from 0 through n-1.

EV[] is the output matrix of eigenvectors stored columnwise. That is, the elements of each eigenvector appear in sequential memory order. The jth element of the ith eigenvector is EV[n*i+j] = EV[i][j].

E[] is the output matrix of eigenvalues. The ith element of E corresponds to the ith eigenvector (the ith row of EV).

On output, the matrix A will have been diagonalized and its original contents are destroyed.

ACCURACY:

The error is controlled by an internal parameter called RANGE which is set to 1e-10. After diagonalization, the off-diagonal elements of A will have been reduced by this factor.

ERROR MESSAGES:

None.

JavaScript:

```
function test_eigens()
{
        var A = [
                        [0.1,0.2,0.3,0.4],
                        [0.5,0.6,0.7,0.8],
                        [0.9,0.8,0.7,0.6],
                        [0.5,0.4,0.3,0.2]
```

```
        ];
        var EV = new Array();
        var E = new Array();
        var N = 4;
        Session.Output("calling cephes.eigens( A, EV, E, N) where:");
        Session.Output(" A is NxN input matrix and N = " + N);
        printMatrix("A",A,N,N);
        cephes.eigens(A, EV, E, N);
        Session.Output(" EV is matrix of " + dimensionsOfArray(EV));
        printMatrix("Y",EV,N,N);
        Session.Output(" ");
        Session.Output(" E is matrix of " + dimensionsOfArray(E));
        printMatrix("Y",E,N,N);
        Session.Output(" ");
}


function printMatrix(name, M, rows, cols)
{
        for(var r = 0; r < rows; r++)
        {
                for(var c = 0; c < cols; c++)
                {
                        Session.Output(name + "[" + r + "][" + c + "] = " +M[r][c]);
                }
        }
}

var str="";

function dimensionsOfArrayX(v)
{
        str += v.length;
        if(v.length)
        {
                var e = v[0];
                if(Array.isArray(e))
                {
                        str += " x ";
                        dimensionsOfArrayX(e);
                }
        }
```

```
}
function dimensionsOfArray(v)
{
	str = "";
	dimensionsOfArrayX(v);
	return str;
}
```

数值集成

- simpsn - 辛普森法则

simpsn

Simpson Numerical Integration

SYNOPSIS

```
double simpsn( f, delta )
double f[];          /* tabulated function */
double delta;        /* spacing of arguments */
double simpsn( f, delta );
```

DESCRIPTION

Numerical integration of function tabulated at equally spaced arguments
Uses 8th order Cote integration formula.

复杂算法

- cadd - 复杂的加法
- csub - 减法
- cmul - 乘法
- cdiv - 除法
- cabs - 绝对价值
- csqrt -根

cadd

Addition.

SYNOPSIS:

```
typedef struct {
    double r;    real part
    double i;    imaginary part
   }cmplx;

cmplx *a, *b, *c;

cadd(a, b, c);    c = b + a
```

DESCRIPTION:

```
  c.r  =  b.r + a.r
  c.i  =  b.i + a.i
```

ACCURACY:

In DEC arithmetic, the test (1/z) * z = 1 had peak relative error 3.1e-17, rms 1.2e-17. The test (y/z) * (z/y) = 1 had peak relative error 8.3e-17, rms 2.1e-17.

Tests in the rectangle {-10,+10}:

			Relative error:	
arithmetic	function	# trials	peak	rms
DEC	cadd	10000	1.4e-17	3.4e-18
IEEE	cadd	100000	1.1e-16	2.7e-17
DEC	csub	10000	1.4e-17	4.5e-18
IEEE	csub	100000	1.1e-16	3.4e-17
DEC	cmul	3000	2.3e-17	8.7e-18
IEEE	cmul	100000	2.1e-16	6.9e-17
DEC	cdiv	18000	4.9e-17	1.3e-17
IEEE	cdiv	100000	3.7e-16	1.1e-16

JavaScript:

```
var a = {"r":0.5,"i",0.5};
```

```
var b = {"r":0.5,"i",0.5};
var c = cephes.cadd(a,b);
Session.Output("c.r=" + c.r + ", c.i=" + c.i);
```

csub

Subtraction.

SYNOPSIS:

```
typedef struct {
    double r;    real part
    double i;    imaginary part
   }cmplx;

cmplx *a, *b, *c;
csub(a, b, c);    c = b - a
```

DESCRIPTION:

```
  c.r  =  b.r - a.r
  c.i  =  b.i - a.i
```

ACCURACY:

In DEC arithmetic, the test (1/z) * z = 1 had peak relative error 3.1e-17, rms 1.2e-17. The test (y/z) * (z/y) = 1 had peak relative error 8.3e-17, rms 2.1e-17.

Tests in the rectangle {-10,+10}:

			Relative error:	
arithmetic	function	# trials	peak	rms
DEC	cadd	10000	1.4e-17	3.4e-18
IEEE	cadd	100000	1.1e-16	2.7e-17
DEC	csub	10000	1.4e-17	4.5e-18
IEEE	csub	100000	1.1e-16	3.4e-17
DEC	cmul	3000	2.3e-17	8.7e-18
IEEE	cmul	100000	2.1e-16	6.9e-17
DEC	cdiv	18000	4.9e-17	1.3e-17
IEEE	cdiv	100000	3.7e-16	1.1e-16

JavaScript:

```
var a = {"r":0.5,"i",0.5};
var b = {"r":0.5,"i",0.5};
var c = cephes.csub(a,b);
```

```
Session.Output("c.r=" + c.r + ", c.i=" + c.i);
```

cmul

Multiplication.

SYNOPSIS:

```
typedef struct {
    double r;    real part
    double i;    imaginary part
   }cmplx;

cmplx *a, *b, *c;

cmul(a, b, c);    c = b * a
```

DESCRIPTION:

```
  c.r  =  b.r * a.r  -  b.i * a.i
  c.i  =  b.r * a.i  +  b.i * a.r
```

ACCURACY:

In DEC arithmetic, the test (1/z) * z = 1 had peak relative error 3.1e-17, rms 1.2e-17. The test (y/z) * (z/y) = 1 had peak relative error 8.3e-17, rms 2.1e-17.

Tests in the rectangle {-10,+10}:

			Relative error:	
arithmetic	function	# trials	peak	rms
DEC	cadd	10000	1.4e-17	3.4e-18
IEEE	cadd	100000	1.1e-16	2.7e-17
DEC	csub	10000	1.4e-17	4.5e-18
IEEE	csub	100000	1.1e-16	3.4e-17
DEC	cmul	3000	2.3e-17	8.7e-18
IEEE	cmul	100000	2.1e-16	6.9e-17
DEC	cdiv	18000	4.9e-17	1.3e-17
IEEE	cdiv	100000	3.7e-16	1.1e-16

JavaScript:

```
var a = {"r":0.5,"i",0.5};
var b = {"r":0.5,"i",0.5};
var c = cephes.cmul(a,b);
```

```
Session.Output("c.r=" + c.r + ", c.i=" + c.i);
```

cdiv

Division.

SYNOPSIS:

```
typedef struct {
     double r;     real part
     double i;     imaginary part
    }cmplx;

cmplx *a, *b, *c;

cdiv(a, b, c);     c = b / a
```

DESCRIPTION:

```
   d    =  a.r * a.r  +  a.i * a.i
   c.r  = (b.r * a.r  + b.i * a.i)/d
   c.i  = (b.i * a.r  -  b.r * a.i)/d
```

ACCURACY:

In DEC arithmetic, the test (1/z) * z = 1 had peak relative error 3.1e-17, rms 1.2e-17. The test (y/z) * (z/y) = 1 had peak relative error 8.3e-17, rms 2.1e-17.

Tests in the rectangle {-10,+10}:

			Relative error:	
arithmetic	function	# trials	peak	rms
DEC	cadd	10000	1.4e-17	3.4e-18
IEEE	cadd	100000	1.1e-16	2.7e-17
DEC	csub	10000	1.4e-17	4.5e-18
IEEE	csub	100000	1.1e-16	3.4e-17
DEC	cmul	3000	2.3e-17	8.7e-18
IEEE	cmul	100000	2.1e-16	6.9e-17
DEC	cdiv	18000	4.9e-17	1.3e-17
IEEE	cdiv	100000	3.7e-16	1.1e-16

JavaScript:

```
var a = {"r":0.5,"i",0.5};
```

```
var b = {"r":0.5,"i",0.5};
var c = cephes.cdiv(a,b);
Session.Output("c.r=" + c.r + ", c.i=" + c.i);
```

cabs

Complex absolute value.

SYNOPSIS:

```
double cabs();
cmplx z;
double a;
a = cabs(&z);
```

DESCRIPTION:

If z = x + iy

then

a = sqrt(x**2 + y**2).

Overflow and underflow are avoided by testing the magnitudes of x and y before squaring. If either is outside half of the floating point full scale range, both are rescaled.

ACCURACY:

			Relative error:	
arithmetic	domain	# trials	peak	rms
DEC	-30,+30	30000	3.2e-17	9.2e-18
IEEE	-10,+10	100000	2.7e-16	6.9e-17

JavaScript:

```
var z = {"r":3.14,"i":3.14};
var a = cephes.cabs(z);
```

where a is an object of schema

```
{
  "r" : double,
  "i" : double
}
```

csqrt

Complex square root.

SYNOPSIS:

void csqrt();
cmplx z, w;

csqrt(&z, &w);

DESCRIPTION:

If z = x + iy, r = |z|, then

$$\text{Im } w = [(r - x)/2]^{1/2},$$

Re w = y / 2 Im w.

Note that -w is also a square root of z. The root chosen is always in the upper half plane.

Because of the potential for a cancellation error in r - x, the result is sharpened by doing a Heron iteration (see sqrt) in complex arithmetic.

ACCURACY:

		Relative error:		
arithmetic	domain	# trials	peak	rms
DEC	-10,+10	25000	3.2e-17	9.6e-18
IEEE	-10,+10	100000	3.2e-16	7.7e-17

JavaScript:

var x = {"r":4.5,"i":3.14} ;
var a = cephes.csqrt(x);

returns a, complex object of schema
{
"r" : double,

"i": double

}

复指数和三角函数

- cexp - 指数
- clog - 对数
- ccos - 余弦
- cacos - 反cosine
- csin - Sine
- casin - 反正sine
- ctan - 正切
- catan - 反正切
- ccot - 余切

cexp

Complex exponential function.

SYNOPSIS:

```
void cexp();
cmplx z, w;

cexp(&z, &w);
```

DESCRIPTION:

Returns the exponential of the complex argument z into the complex result w.

If

$z = x + iy$,

$r = \exp(x)$,

then

$w = r \cos y + i\, r \sin y$.

ACCURACY:

		Relative error:		
arithmetic	domain	# trials	peak	rms
DEC	-10,+10	8700	3.7e-17	1.1e-17
IEEE	-10,+10	30000	3.0e-16	8.7e-17

clog

Complex natural logarithm.

SYNOPSIS:

```
void clog();
cmplx z, w;

clog(&z, &w);
```

DESCRIPTION:

Returns a complex logarithm to the base e (2.718...) of the complex argument x.

If z = x + iy, r = sqrt(x**2 + y**2),
then
    w = log(r) + i arctan(y/x).

The arctangent ranges from -PI to +PI.

ACCURACY:

		Relative error:		
arithmetic	domain	# trials	peak	rms
DEC	-10,+10	7000	8.5e-17	1.9e-17
IEEE	-10,+10	30000	5.0e-15	1.1e-16

Larger relative errors can be observed for z near 1 +i0. In IEEE arithmetic the peak absolute error is 5.2e-16, rms absolute error 1.0e-16.

CCOS

Complex circular cosine.

SYNOPSIS:

```
void ccos();
cmplx z, w;

ccos(&z, &w);
```

DESCRIPTION:

If

$$z = x + iy,$$

then

$$w = \cos x \cosh y - i \sin x \sinh y.$$

ACCURACY:

		Relative error:		
arithmetic	domain	# trials	peak	rms
DEC	-10,+10	8400	4.5e-17	1.3e-17
IEEE	-10,+10	30000	3.8e-16	1.0e-16

cacos

Complex circular arc cosine.

SYNOPSIS:

```
void cacos();
cmplx z, w;

cacos(&z, &w);
```

DESCRIPTION:

w = arccos z = PI/2 - arcsin z.

ACCURACY:

		Relative error:		
arithmetic	domain	# trials	peak	rms
DEC	-10,+10	5200	1.6e-15	2.8e-16
IEEE	-10,+10	30000	1.8e-14	2.2e-15

csin

Complex circular sine.

SYNOPSIS:

```
void csin();
cmplx z, w;

csin(&z, &w);
```

DESCRIPTION:

If

$$z = x + iy,$$

then

$$w = \sin x \cosh y + i \cos x \sinh y.$$

ACCURACY:

	Relative error:			
arithmetic	domain	# trials	peak	rms
DEC	-10,+10	8400	5.3e-17	1.3e-17
IEEE	-10,+10	30000	3.8e-16	1.0e-16

casin

Complex circular arc sine.

SYNOPSIS:

```
void casin();
cmplx z, w;

casin(&z, &w);
```

DESCRIPTION:

Inverse complex sine:

$$w = -i \log(iz + \sqrt{1 - z^2}).$$

ACCURACY:

	Relative error:			
arithmetic	domain	# trials	peak	rms
DEC	-10,+10	10100	2.1e-15	3.4e-16
IEEE	-10,+10	30000	2.2e-14	2.7e-15

Larger relative error can be observed for z near zero. Also tested by csin(casin(z)) = z.

ctan

Complex circular tangent.

SYNOPSIS:

```
void ctan();
cmplx z, w;

ctan(&z, &w);
```

DESCRIPTION:

If

$$z = x + iy,$$

then

$$w = \frac{\sin 2x + i \sinh 2y}{\cos 2x + \cosh 2y}.$$

On the real axis the denominator is zero at odd multiples of PI/2. The denominator is evaluated by its Taylor series near these points.

ACCURACY:

		Relative error:		
arithmetic	domain	# trials	peak	rms
DEC	-10,+10	5200	7.1e-17	1.6e-17
IEEE	-10,+10	30000	7.2e-16	1.2e-16

Also tested by ctan * ccot = 1 and catan(ctan(z)) = z.

catan

Complex circular arc tangent.

SYNOPSIS:

```
void catan();
cmplx z, w;

catan(&z, &w);
```

DESCRIPTION:

If

$$z = x + iy,$$

then

$$\mathrm{Re}\ w = -\frac{1}{2}\arctan\left(\frac{2x}{(1 - x^2 - y^2)}\right) + k\,\mathrm{PI}$$

$$\mathrm{Im}\ w = -\frac{1}{4}\log\left(\frac{(x^2 + (y+1)^2)}{(x^2 + (y-1)^2)}\right)$$

Where k is an arbitrary integer.

ACCURACY:

		Relative error:		
arithmetic	domain	# trials	peak	rms
DEC	-10,+10	5900	1.3e-16	7.8e-18
IEEE	-10,+10	30000	2.3e-15	8.5e-17

The check catan(ctan(z)) = z, with |x| and |y| < PI/2, had peak relative error 1.5e-16, rms relative error 2.9e-17. See also clog().

ccot

Complex circular cotangent.

SYNOPSIS:

```
void ccot();
cmplx z, w;

ccot(&z, &w);
```

DESCRIPTION:

If

$$z = x + iy,$$

then

$$w = \frac{\sin 2x - i \sinh 2y}{\cosh 2y - \cos 2x}.$$

On the real axis, the denominator has zeros at even multiples of PI/2. Near these points it is evaluated by a Taylor series.

ACCURACY:

	Relative error:			
arithmetic	domain	# trials	peak	rms
DEC	-10,+10	3000	6.5e-17	1.6e-17
IEEE	-10,+10	30000	9.2e-16	1.2e-16

Also tested by ctan * ccot = 1 + i0.

错误

Printing an error message

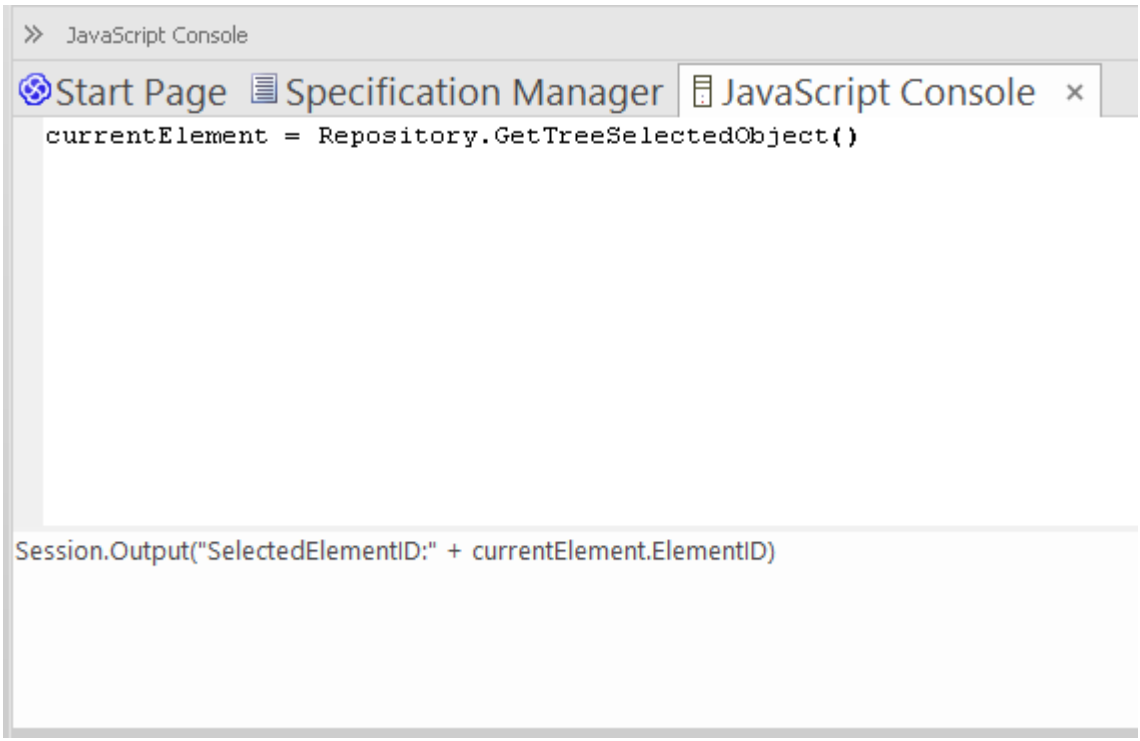
```
var cephes_errors = [
"'unknown'",      /* error code 0 */
"'domain'",       /* error code 1 */
"'singularity'",  /* et seq.      */
"'overflow'",
"'underflow'",
"'total loss of precision'",
"'partial loss of precision'" ];

function printError()
{
        var er = cephes.geterror();
        if(er>0)
        {
                Session.Output( "cephes error " + err + " " + cephes_errors[er]);
        }
}
```

Testing for error

```
if(cephes.inerror())
{
   printError();
}
```

JavaScript Console

JavaScript控制台是一个命令行解释器，它接受单行JavaScript命令，一次执行一个。您在屏幕底部的文本输入面板中键入命令，当您按 Enter 键执行命令时，它会与命令的任何输出一起添加到上方的输出窗口。考虑这个例子。



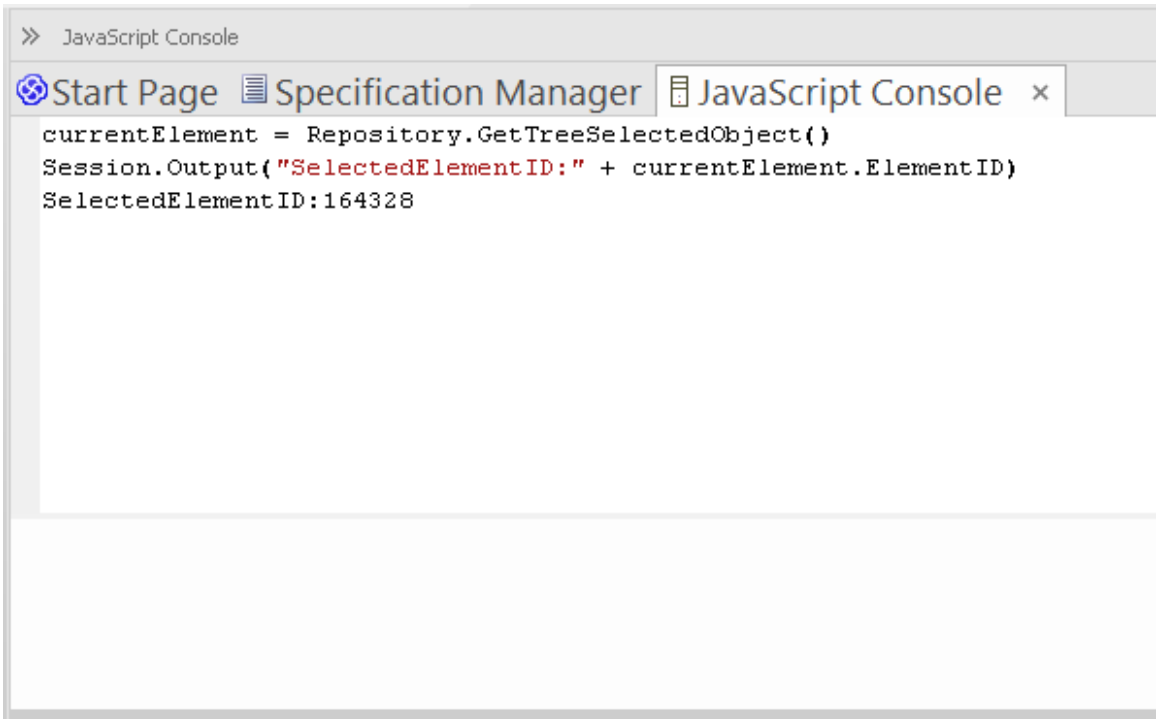
在底部面板中，用户输入了：

currentElement =存储库.GetTreeSelectedObject()

他们按下了 Enter 键，该命令已显示在顶部面板中。然后，用户在下部面板中键入：

Session.输出("选中的ElementID:"+currentElement.ElementID)

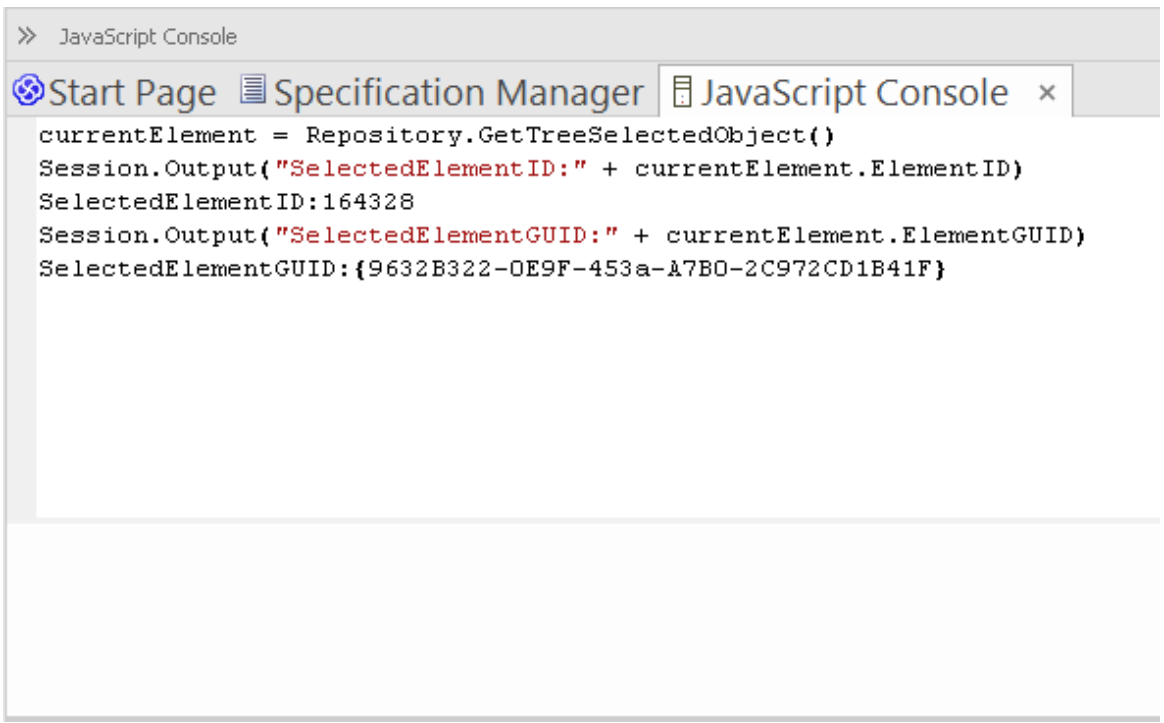
当他们按下 Enter 键时，控制台会显示此命令和命令的输出。



然后用户输入第三个命令：

Session.输出("SelectedElementGUID:" + currentElement.ElementGUID)

这将导致此处显示的输出，在上面板中：



此特征在Enterprise Architect的企业统一版和终极版中可用。

访问

功能区	特定> 工具 > JavaScript

	仿真> 控制台> JavaScript

控制台命令

控制台命令前面有 !角色并指示控制台执行操作。

可用的控制台命令包括：

- !clear - 清除控制台显示
- !save - 将控制台显示保存到文件中
- !help - 打印命令列表
- !close - 关闭控制台
- !include <scriptname> - 执行指定的脚本项； scriptname 的格式为 GroupName.ScriptName（名称中允许使用空格）
- ? - 列出命令（与!help相同）
- ? <变量或函数名> - 输出值。

要在 控制台”选项卡本身上列出这些命令，请键入？在下部面板中（没有前面的！字符），然后按 Enter 键。

如果你打算执行脚本，你可能脚本打开脚本库（脚本窗口），这样你就可以看到可以调用的脚本。选择 特定>工具>脚本库”功能区选项。

控制台窗口

控制台窗口是一个命令行解释器，您可以通过它快速启用脚本引擎并输入命令以对脚本进行操作（JScript、JavaScript和 VBScript）。

您可以通过仿真和特定功能区打开JavaScript控制台窗口。您可以通过特定功能区打开 VBScript 和 JScript 的控制台窗口。

对于这三种脚本语言，您可以在窗口底部的字段中键入命令；当您按下 Enter 键时，脚本控制台将执行命令并立即显示任何输出。控制台命令在*JavaScript Console*帮助主题中进行了描述。

访问

功能区	特定> 工具 > JavaScript 特定> 工具 > VBScript 特定> 工具 > JScript 仿真> 控制台> JavaScript

注记

- 此功能在企业统一版和终极版中可用

求解器接口

Solvers接口使您能够在JavaScript中调用一组命令，这些命令定义和制定 Solver类以对数据执行数学运算。Solver类的原理函数是在模拟过程中提供与 MATLAB 和 Octave 等外部工具的集成，并在 Octave 或 MATLAB 中公开结果，或者将它们带回Enterprise Architect以在那里表示，也许在动态图表中。更一般地，求解器接口可用于基于模型的插件和自定义脚本。

要从 Octave 或 MATLAB 调用函数，您需要熟悉相应产品库中可用的函数，如产品文档中所述。

求解器构造函数

构造器	描述
Solver(string solverName)	创建一个连接到指定助手应用程序的新实例的新求解器。

求解器方法

方法	描述
get(string name)	从 Solver 环境中检索命名值。
set(string name, object value)	为 Solver 环境中的命名变量分配一个新值。
exec(string name, string arguments, int returnValues)	执行一个命名函数。实际功能将取决于正在使用的 Solver 的类型。

脚本编辑

使用脚本编辑器可以对打开的脚本文件执行一些操作，例如：

- 保存对当前脚本的更改
- 以不同的名称保存当前脚本
- 运行脚本
- 调试脚本
- 停止执行脚本
- 在系统输出视图的 脚本”选项卡中查看脚本输出

该编辑器基于应用程序工作区中的通用代码编辑器并提供其功能。

访问

功能区	特定>工具>脚本库>展开脚本组右键【脚本名称】>编辑脚本 特定>工具>脚本库>展开脚本组，双击【脚本名称】

功能

功能	细节
脚本Objects	Enterprise Architect通过提供内置对象增加了编辑器脚本语言的可用功能和特征；这些类型库要么提供智能感知用于编辑目的，要么是运行时对象，提供对库中描述的类型的对象的访问。 可用的智能感知脚本对象有： • EA • 数学库 • 系统 运行时脚本对象是： • 存储库（类型：IDualRepository， EA.Repository的一个实例， Enterprise Architect Automation接口） • 数学（类型：IMath，MathLib 的一个实例；它公开了 Cephes 数学库中的函数以在脚本中使用） • Session（类型：ISession，系统实例）
脚本编辑智能感知(需要Syntax)	智能感知是“脚本器”中的“脚本脚本”，不仅在脚本引擎中提供，而且其智能感知的功能是为内置脚本引擎提供的。 对于智能感知类型；附加的Enterprise Architect脚本对象（如所列），您必须根据指定的语法声明变量不必使用此语法来正确执行脚本，它仅存在于为项目显示正确的智能感知。 语法可以在例如： Dim e as EA.元素 然后，当您键入时，在本例中为 e.，编辑器会显示成员函数列表和 e 类型的属性。

	您选择其中之一来完成该行脚本；因此，您可以键入： VBTrace（例如。 当您键入句点时，编辑器会显示相应的列表，您可以双击，例如，摘要；这被插入到该行中，然后您继续键入或选择语句的其余部分，在这种情况下添加结束空格和括号： VBTrace(e.Abstract)
击键	在脚本编辑器或控制台中，智能感知显示在这些按键上。 • 按 。 （句点）在项目之后列出该项目类型的任何成员 • 按 Ctrl+一个单词以列出任何以在按下键的位置以string开头的名称的智能感知项目 • 当不在单词上按 Ctrl+ 显示任何可用的顶级智能感知项目以及已描述的智能感知项目 - 这些是当前脚本语言的任何内置方法和属性的内置对象
包含脚本库	*Include*语句 (!INC) 允许脚本引用由脚本窗口中可访问的另一个脚本定义的常量、函数和变量。包含语句通常用在脚本的开头。 要包含脚本库，请使用以下语法： !INC [脚本组名称].[脚本名称] 例如： !INC 本地脚本.EAConstants-VBScript
使用内置数学函数	通过使用内置的 Maths object ，在脚本编辑器中可以使用各种数学函数。 您可以在脚本编辑器中通过键入 "Maths"和句点来访问 Maths object 。 智能特征显示智能感知数学库提供的可用数学函数列表。例如： Session.输出"9的平方根是" &根. sqrt (9) Session.输出"2^10 = " & 数学。 pow (2,10) 数学object在Enterprise Architect的统一版和终极版中可用。
使用 COM / ActiveX 对象	VBScript、JScript 和JavaScript都可以创建并使用 ActiveX/COM 对象。这可以帮助您使用外部库，或与Enterprise Architect外部的其他应用程序进行交互。例如，脚本脚本类可以用来读写本地机器上的文件。每种语言创建新object的语法略有不同，如以下示例所示： VB脚本： set脚本= CreateObject("脚本.FileSystemObject") 脚本： 脚本= new ActiveXObject("脚本.FileSystemObject"); JavaScript ： fsObject = new脚本("脚本.FileSystemObject");
将JavaScript与进程外 COM 服务器一起使用	Enterprise Architect中的JavaScript用户可以访问进程外 COM 服务器。该应用程序必须在机器上注册为提供本地服务器支持。创建或获取对进程外服务器的引用的语法是： var server = new COMObject(*progID* , true); 其中*progID*是 COM 组件的注册程序 ID（例如，“Excel.Application”）。
系统脚本库	当Enterprise Architect安装在您的系统上时，它包括一个默认脚本库，该库提供了许多有用的脚本函数，从简单的string函数到用于定义您自己的 CSV 或 XMI 导入和导出的函数。 要使用脚本库，您必须在 "MDG 技术"对话框（"特定>技术>管理技术"功能区选项）中启用它。

	滚动浏览技术列表，然后选中 "EAScriptLib"对应的 "启用"复选框。

注记

- 统一版和企业版中有脚本编辑终极
- Enterprise Architect脚本支持声明变量以匹配Enterprise Architect类型；这使编辑器能够展示智能感知，但对于执行脚本不是必需的

会话物件

Session 运行时object提供跨所有脚本语言的通用输入/反馈机制，允许访问系统类型库中描述的类型的对象。它可以通过脚本窗口在Enterprise Architect运行任何脚本。

属性

属性	细节
属性	• 用户名- 返回当前的 windows 用户名（只读） • 版本- 返回此object的版本（只读）
方法	• Input(string Prompt) - 显示一个对话框，提示用户输入一个值；返回用户输入的string值 • 输出（ string输出） - 将文本写入当前默认输出位置；期间： - 正常脚本执行，输出写入系统输出窗口的 脚本”选项卡 -脚本调试，输出写入调试窗口 -使用脚本Console，输出写入Console • Prompt(string Prompt, long PromptType) - 显示一个包含指定提示文本和按钮类型的模式对话框；返回与用户单击的按钮对应的 PromptResult”值
提示类型值	• 提示OK = 1 • 提示YESNO = 2 • 提示YESNOCANCEL = 3 • 提示OKCANCEL = 4
提示结果值	• 结果OK = 1 • 结果取消 = 2 • 结果是 = 3 • 结果编号 = 4
Session.Prompt示例	(VBScript) If (Session.Prompt("Continue?", promptYESNO) = resultYes) 那么...

工作流程

工作流根据您的模型中的政策和程序验证用户的工作和行动，为应用公司政策和加强项目开发指南提供了一种强大的方法。

项目管理员可以编写工作流来管理用户与模型交互的方式，例如管理安全性、员工合规性和模型访问，以及监控用户所做的更改。管理员还可以使用工作流来控制用户更改模型元素的能力，考虑到诸如访问权限、组成员资格甚至提议更改的价值等因素。

工作流程的应用

考虑	描述
项目治理	良好的公司治理依赖于书面和透明的项目开发指南和公司政策。 A对适当的政策和程序知之甚少并且没有正确遵循，项目可能会受到影响——人为错误和从开发人员的不充分合规中恢复的成本可能会阻碍有效的治理。
政策、过程与发展	公司政策和程序可以与开发流程集成，以管理工作流程、确定访问权限、扩展基于角色的安全权限并响应属性更改事件。 这种方法降低了合规成本，增强了协作开发，并让您确信项目在第一次就正确开发。 开发团队可以遵守管理模型验证、变更管理、访问控制和一般开发原则的最佳实践指南。

工作流程选项

使用 Workflow 脚本有两个选项：

- VBScript
- JavaScript 。

VBScript 是较旧的版本，仅限于一系列命令。 JavaScript版本是更新的版本，允许完全访问所有自动化功能。

JavaScript记录在*EA_Connect*和*Workflow*插件

下插件

事件帮助主题。

VBScript 记录在脚本脚本函数帮助主题下。

注记

- Enterprise Architect的企业统一版和终极版中提供了工作流程

工作流脚本函数

注记：从Enterprise Architect 15.0 版开始，VBScript 工作流脚本可供使用但已弃用。您现在可以使用Enterprise Architect插件

模型事件 EA_Connect 以响应 Workflow插件

事件，具有更广泛的特征并且不依赖于 Visual Basic。

工作流脚本在脚本窗口中创建，工作流组类型为 VBScripts。它们由Enterprise Architect工作流引擎执行，以管理用户输入。

您可以使用一系列函数和数据结构来开发您的脚本。

访问

使用此处概述的方法之一打开脚本窗口，然后单击 新建组”按钮创建一个新的工作流脚本组，然后单击 新建脚本”按钮创建一个新脚本。

功能区	特定> 工具 >脚本

工作流函数和数据结构

函数	描述
脚本实现	启动模型时，工作流引擎将使用当前用户和组成员身份进行初始化；该信息决定了谁可以访问和修改给定模型的某些部分。 当发生选定的事件时，脚本引擎将以包括作者的名称和访问权限以及元素名和版本详细信息在内的值初始化。 工作流脚本执行管理变更管理、访问控制和模型验证的规则；如果用户尝试违反公司政策进行更改，脚本将拒绝更新。 通知用户验证失败的原因并记录活动。 这些提醒有助于强化公司政策、减少人为错误并为管理层提供有价值的项目反馈。
用户输入函数	这些是Enterprise Architect调用来验证和控制用户输入的函数。 对于Enterprise Architect调用的每个函数，都填充了一组对象。
创建搜索的函数	Enterprise Architect调用这些函数来创建包含用户任务的搜索。
工作流数据结构Enterprise Architect填充	这些是Enterprise Architect填充的工作流数据结构对象。
您填写的工作流数据结构	这些是您可以填充的工作流数据结构对象。
你函数	这些是Enterprise Architect提供给您调用的功能。

注记

- 如果您对脚本窗口中列出的工作流脚本进行了更改，请单击脚本窗口工具栏中的刷新脚本按钮以重新加载更改后的脚本
- Workflow脚本在企业统一版和Enterprise Architect终极版中可用
- 工作流脚本需要启用用户安全才能函数
- 您需要 管理工作流程"权限才能开发和管理工作流程脚本

函数-验证和控制用户输入

Enterprise Architect调用许多函数来验证和控制用户输入。对于每个函数，都会填充一组对象。

验证/控件用户

函数	行动
AllowPhaseUpdate（旧值， 新值）	验证用户对相所做的更改。 返回价值： • True允许此用户进行此更改 • False表示不允许更改并恢复到以前的值
AllowStatusUpdate（旧值， 新值）	验证用户对状态所做的状态。 返回价值： • True允许此用户进行此更改 • False表示不允许更改并恢复到以前的值
AllowTagUpdate（标签名称， 旧值， 新值）	验证用户对标记值所做的更改。 返回价值： • True允许此用户进行此更改 • False表示不允许更改并恢复到以前的值
AllowVersionUpdate（旧值， 新值）	验证用户对版本所做的更改。 返回价值： • True允许此用户进行此更改 • False表示不允许更改并恢复到以前的值
可以编辑阶段（ ）	启用或禁用编辑相的控件 返回价值： • True允许此用户通过启用控件进行更改 • False通过禁用控件完全禁用编辑此属性
可以编辑状态（ ）	启用或禁用用于编辑状态的控件。 返回价值： • True允许此用户通过启用控件进行更改 • False通过禁用控件完全禁用编辑此属性
CanEditTag（标签名称）	启用或禁用编辑标记值控件。 返回价值： • True允许此用户通过启用控件进行更改 • False通过禁用控件完全禁用编辑此属性
可以编辑版本（ ）	启用或禁用用于编辑版本的控件。 返回价值：

	• True允许此用户通过启用控件进行更改 • False通过禁用控件完全禁用编辑此属性
OnPreNewElement（元素类型， 元素刻板印象）	允许或禁止创建指定的元素。 返回价值： • True允许此用户创建元素/连接器 • False以防止此用户创建元素
OnPreNewConnector（连接器类型， 连接器子类型， 连接器原型）	允许或禁止创建指定的连接器。 返回价值： • True允许此用户创建元素/连接器 • False以防止此用户创建元素
PreAllowPhaseUpdate（旧值， 新值）	确定验证此更改所需的信息。 返回值：以分号分隔的附加数据列表，以验证此更改。 支持的数据类型： • 测试 - 在 WorkflowContext object中填充测试数组
PreAllowStatusUpdate（旧值， 新值）	确定验证此更改所需的信息。 返回值：以分号分隔的附加数据列表，以验证此更改。 支持的数据类型： 测试 - 在 WorkflowContext object中填充测试数组
PreAllowTagUpdate（标签名称， 旧值， 新值）	确定验证此更改所需的信息。 返回值：以分号分隔的附加数据列表，以验证此更改。 支持的数据类型： 测试 - 在 WorkflowContext object中填充测试数组
PreAllowVersionUpdate（旧值， 新值）	确定验证此更改所需的信息。 返回值：以分号分隔的附加数据列表，以验证此更改。 支持的数据类型： 测试 - 在 WorkflowContext object中填充测试数组

函数- 使用用户任务创建搜索

Enterprise Architect调用这些函数来创建包含用户任务的搜索。

函数

函数	行动
获取工作流任务	描述该用户必须运行的搜索。 返回值：忽略

填充的工作流数据结构

这些是Enterprise Architect填充的工作流数据结构（对象）。

数据结构

工作流数据结构	描述
工作流用户	该object提供有关当前登录到模型的用户的信息。 它在 Enterprise Architect 调用任何函数之前由Enterprise Architect Enterprise Architect填充；它有以下属性： • 用户名 - 登录系统的用户名（如果使用窗口身份验证，则与窗口用户名匹配） • 名字 - 在 安全用户”对话框中找到 • 姓氏 - 在 安全用户”对话框中找到 • 全名 - <Firstname> <Surname> 的组合（ Enterprise Architect用于“作者”字段和类似字段的形式） • 部门 - 用户工作的部门，可在 安全用户”对话框中找到 调用：此object调用 IsMemberOf(GroupName)函数。
工作流上下文	此object提供有关当前时间中的object的上下文。 它在除运行之外的任何搜索运行之前由Enterprise Architect填充；它有以下属性： • MetaType - 当前object的类型，可以是Enterprise Architect核心类型或配置文件指定的元类型 • 名称- 在object 属性”对话框中找到 • 状态- 在object 属性”对话框中找到 • 相在object 属性”对话框中找到 • -版本- 在object 属性”对话框中找到 • 构造型- 应用于此object的构造型的字符串数组 • 标签 - 一个标记值数组，提供： -名称-标记值名称 - 值 -标记值值 • 测试 - 一系列测试；仅在 PreAllow* 调用指定需要测试后的 Allow* 调用期间填充；提供这些详细信息，如测试案例窗口中所示： -名称 -状态 - 由.....运营 - 通过检查 - 测试类 - 测试类型 Calls：这个object调用标签值(TagName)函数。

函数

函数	行动
IsMemberOf(组名)	选择当前用户的组成员身份。 返回值：如果当前用户是具有指定名称的组的成员，则返回值True 。
标签值(TagName)	从命名标签中获取值。 返回值：返回该名称的第一个标记值的值，如果不存在该名称的标记值，则返回空string 。

您填写的工作流数据结构

这些是您可以填写的工作流数据结构（对象）。

数据结构

工作流数据结构	描述
工作流程状态	使用这个数据结构来提供关于object状态的信息。 • LogEntry - 设置为True或False以指示是否应记录log项 • 原因 - 指明应该在log中记录什么原因 • 行动——指示如何显示log消息；有效值为：MessageBox、StatusBar 和输出（默认）
工作流程搜索	使用这个数据结构来提供一个搜索数组。 使用Redim WorkflowSearches(x) 指定提供的搜索数量。 每个搜索都有以下属性： • 名称- 此搜索的名称 • 组 - 此搜索应出现在 搜索”组合框中的组的名称 • ID - 此搜索的GUID • 任务 - 此搜索查找的任务数组；一个条目描述了如何找到满足特定任务所需的所有对象： -名称-模型搜索中显示的任务名称 看法;默认情况下，工作流搜索按此字段分组 - 条件 - 条件数组，所有条件都必须匹配 包含在此任务中的object ；条件是比较 单个字段到一个值： - 列 - 字段的名称 - 运算符 - 运算符类型，或者 =（仅提供匹配值） 或 <>（仅提供不匹配的值） - 值 - 如果它包含逗号，则string被视为 要比较的逗号分隔值列表； 否则， string是要比较的单个值

您调用的函数

不明确的

函数

函数	行动
新搜索（名称、组、guid、任务计数）	不明确的
新任务（名称，条件计数）	不明确的
新条件（列、运算符、值）	不明确的
SetLastError（消息，输出方法）	不明确的

脚本调试

脚本调试有助于模型脚本的开发和维护，并在执行时监控它们的活动。在调试脚本时，您可以：

- 使用脚本器工具栏上的 调试”、 节结束”、 节进入”、 节输出”和 停止脚本”按钮来执行控件流程
- 设置断点、记录标记和跟踪点标记
- 使用调试窗口查看脚本生成的输出
- 使用本地窗口窗口来检查变量的值，包括来自自动化接口的对象
- 使用Record & Analyze 窗口记录脚本执行的序列图

访问

功能区	特定>工具>脚本库>右键【脚本名称】>调试脚本
其它	脚本编辑器窗口工具栏：点击  工具栏图标

开始调试模型脚本

节	行动
1	在脚本编辑器中打开一个模型脚本。
2	在适当的代码行上设置任何断点。
3	单击  调试工具栏图标（调试）。

注记

- 脚本、JScript、 JavaScript支持脚本调试
- VBScript和JScript需要在本地机器上安装微软进程调试管理器；这可以通过各种微软产品获得，包括免费的 微软脚本调试器”
- 断点不为脚本保存，下次打开脚本时不会保留
- 调试时，脚本输出被重定向到调试窗口