



ENTERPRISE ARCHITECT

Série de Guides d'Utilisateur

Scriptant

Author: Sparx Systems

Date: 2022-08-26

Version: 16.0

CRÉÉ AVEC  **ENTERPRISE
ARCHITECT**

Table des Matières

Scriptant	6
Fenêtre Scriptant	10
Propriétés du groupe de scripts	12
JavaScript Math Library	14
Arithmétique et Algébrique	15
sqrt	16
lsqrt	17
cbrt	18
polevl , p1evl	19
chbevl	21
round	22
floor	23
ceil	24
frexp	25
ldexp	26
fabs	27
signbit	28
isnan	29
isfinite	30
poladd	31
polsub	32
polmul	33
poldiv	34
polsbt	35
poleva	36
polclr	37
polmov	38
Exponentiel et Trigonométrie	39
acos	40
acosh	41
asinh	42
atanh	43
asin	44
atan	45
atan2	46
cos	47
cosdg	48
exp	49
exp2	50
exp10	51
cosh	52
sinh	53
tanh	54
log	55
log2	56
log10	57
pow	58

powi	59
sin	60
sindg	61
tan	62
tandg	63
Intégrale Exponentielle	64
expn	65
shichi	66
sici	68
Gamma	70
beta	71
lbeta	72
fac	73
gamma	74
lgam	75
incbet	76
incbi	78
igam	79
igamc	80
igami	81
psi	82
rgamma	84
Fonction d'erreur	85
erf	86
erfc	87
dawsn	88
fresnl	89
Bessel	91
airy	92
j0	94
j1	95
jn	96
jv	97
y0	98
y1	99
yn	100
yv	101
i0	102
i0e	103
i1	104
i1e	105
iv	106
k0	107
k0e	108
k1	109
k1e	110
kn	111
Hypergéométrique	112
hyperg	113
hyp2f1	114
hyp2f0	116

onef2	117
threef0	118
Elliptique	119
ellpe	120
ellie	121
ellpk	122
ellik	124
ellpj	125
Probabilité	126
bdtr	127
bdtrc	129
bdtri	131
chdtr	132
chdtrc	133
chdtri	134
fdtr	135
fdtrc	136
fdtri	138
gdtr	140
gdtrc	141
nbdtr	142
nbdtrc	143
ndtr	144
ndtri	145
pdtr	146
pdtrc	147
pdtri	148
stdtr	149
Divers	151
polylog	152
spence	154
zetac	155
zeta	156
struve	158
Matrice	159
fftr	160
simq	161
minv	162
mmmpy	164
mvmpy	165
mtransp	166
eigens	167
Intégration Numérique	170
simpsn	171
Arithmétique Complexe	172
cadd	173
csub	175
cmul	177
cdiv	179
cabs	181
csqrt	182

Exponentiel Complexe et Trigonométrie	184
cexp	185
clog	186
ccos	187
cacos	188
csin	189
casin	190
ctan	191
catan	192
ccot	193
les erreurs	194
JavaScript Console	195
Fenêtre de la console	198
Interface Solveurs	199
Éditeur de Script	200
Objet de Session	203
Flux de travail	204
Fonctions de script de flux de travail	206
Fonctions - Valider et contrôler l'entrée de l'utilisateur	208
Fonctions - Créer une recherche avec des tâches utilisateur	210
Structures de données de flux de travail remplies	211
Structures de données de workflow que vous remplissez	213
Fonctions que vous appelez	214
Débogage de Script	215

Scriptant



L'environnement de script d' Enterprise Architect est une facilité flexible et facile à utiliser qui facilite à la fois JavaScript et supporte langages de script Microsoft JScript et VBScript. Lorsqu'un script s'exécute, il a accès à un objet 'Référentiel' intégré. À l'aide de cet objet de script, vous pouvez inspecter et/ou modifier par programmation des éléments dans votre modèle actuellement ouvert. Enterprise Architect fournit également des éditeurs riches en fonctionnalité et des outils pour exécuter, déboguer et gérer vos scripts. Les Scripts sont modulaires et peuvent inclure d'autres scripts par leur nom à l'aide de la directive *!include*. Ils peuvent être utilisés pour un large éventail d'objectifs, de la documentation à la validation et à la refactorisation, et ils peuvent être d'une aide considérable pour automatiser les tâches chronophages.

Prise en Support moteur de script

- Mozilla SpiderMonkey [version 1.8]
- Moteur Scriptant Microsoft

Langages de script

- JavaScript
- JScript
- VBScriptName

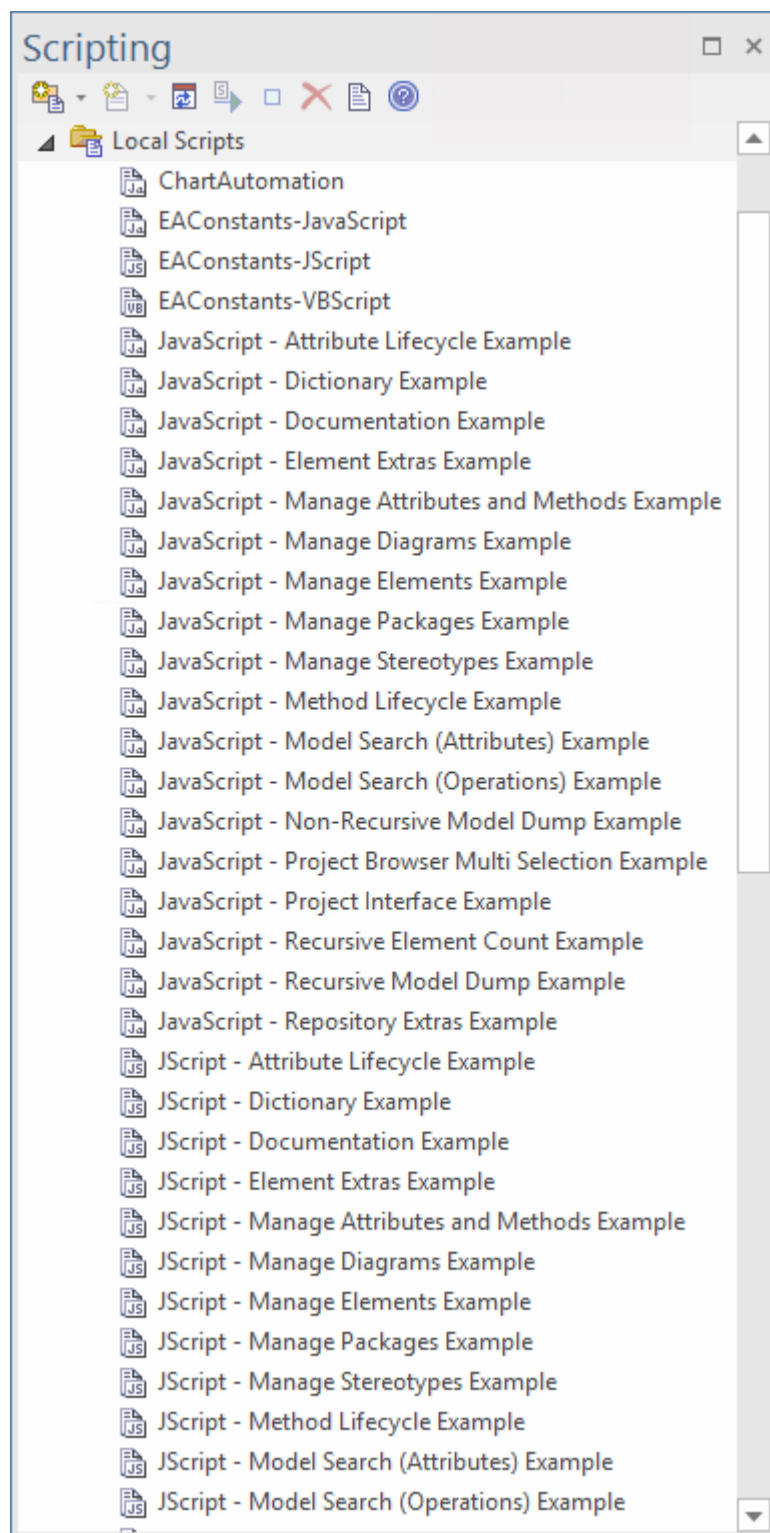
Avantages

- Inspection et rapport sur la composition du modèle et des éléments
- Modification et mise à jour des propriétés des éléments
- Exécution de requêtes pour obtenir des informations étendues sur le modèle
- Modification des mises en page des diagrammes
- Être appelé à partir des gabarits de document de rapport pour remplir les rapports
- Création et mise en œuvre de workflows de processus
- Être inclus dans MDG Technologies pour augmenter les langages spécifiques à un domaine
- Accès étendu de l'UI aux scripts via des menus contextuels
- Rôle Automation Server pour les clients COM in-process et out-of-process (Scriptant est lui-même un exemple de client in-process ; les Add-Ins sont un autre)
- Gouvernance de l'accès aux éléments via la sécurité du workflow
- Modèle de la recherche de modèles

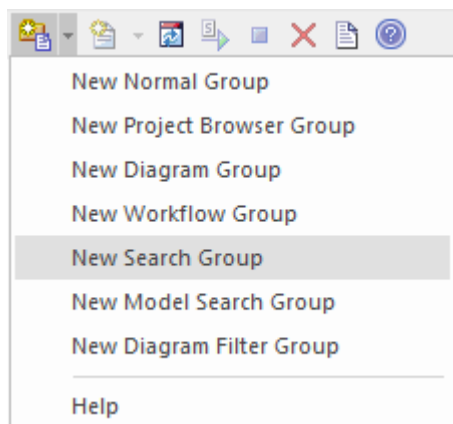
Utiliser Scripts

L'interface de gestion de Scriptant est la fenêtre Scriptant , affichant le Script Tree Vue , que vous utilisez pour révision , créer et éditer des scripts.

Outre les Scripts locaux, qui sont basés sur des fichiers et installés avec Enterprise Architect , tous les autres scripts sont stockés en tant qu'actifs de modèle et peuvent être partagés avec tous les utilisateurs du modèle. Les débogueurs de scripts peuvent vous aider dans le développement de scripts et les éditeurs de scripts peuvent vous fournir des informations sur les interfaces d'automatisation à votre disposition. Vous pouvez analyser l'exécution, par exemple en enregistrant un Séquence de diagramme de l'exécution du script et en arrêtant l'exécution pour afficher les variables locales.



Groupes de scripts



Les Scripts sont gérés et contenus dans des groupes. Chaque groupe a un attribut appelé ' Type '. Cet attribut est utilisé pour aider Enterprise Architect à décider comment et où le script peut être utilisé et à partir de quelles fonctionnalités il doit être rendu disponible. Les propriétés d'un groupe de scripts peuvent être visualisées à partir de son menu contextuel.

Stockage de scripts

Les scripts intégrés sont basés sur des fichiers et sont installés avec Enterprise Architect . Ils apparaissent sous le groupe *Scripts locaux* .

Vous ne pouvez pas modifier ou supprimer des scripts locaux, mais vous pouvez copier le contenu assez facilement.

Les scripts définis par l'utilisateur sont basés sur un modèle et, en tant que tels, peuvent être partagés par une communauté. Ils sont répertoriés dans le groupe auquel ils appartiennent.

Utiliser Solveurs

Anywhere dans Enterprise Architect qui a du code JavaScript , comme dans Simulation , vous pouvez maintenant utiliser une construction JavaScript appelée ' Solveur ' (la classe Solveur) pour s'intégrer à des outils externes et utiliser directement les fonctionnalités de chaque outil pour effectuer simplement et intuitivement fonctions mathématiques et graphiques complexes. Les appels vous aident à échanger facilement des variables entre le moteur JavaScript intégré et chaque environnement. Deux bibliothèques mathématiques prises en charge sont MATLAB et Octave.

Pour utiliser la classe Solveur , vous devez connaître les fonctions disponibles dans votre Bibliothèque Math préférée et les paramètres qu'elles utilisent, comme décrit dans la documentation du produit.

Faisant partie du moteur JavaScript , les classes Solveur sont également immédiatement accessibles aux rédacteurs de Add-In créant des Add-Ins JavaScript basés sur des modèles.

Consultez également les rubriques d' *aide Octave Solveur* , *MATLAB Solveur* et *Solveurs* .

Notes

- Cette facilité est disponible dans les éditions Corporate , Unified et Ultimate
- Si vous avez l'intention d'utiliser la Scriptant de facilité sous Crossover/ WINE , vous devez également installer Internet Explorer version 6.0 ou supérieure

Fenêtre Scriptant

La fenêtre Scriptant est composée d'une barre d'outils et d'une vue de tous les scripts par groupe. Les groupes de scripts et leurs scripts ont également des menus contextuels qui fournissent tout ou partie de ces options :

- Propriétés du groupe - pour afficher ou modifier les propriétés du groupe de scripts dans la boîte de dialogue ' Propriétés du groupe de scripts'
- Exécuter Script - pour exécuter le script sélectionné (ou appuyez sur Ctrl pendant que vous double-cliquez sur le nom du script)
- Script Débogage - pour déboguer le script sélectionné
- Modifier le script - pour mettre à jour le script sélectionné (ou double-cliquez sur le nom du script pour afficher l' ' Éditeur de Script ', qui affiche généralement un gabarit de script , déterminé par le type de groupe d'utilisateurs attribué lors de la création ou sur les ' Propriétés du groupe de scripts' ' dialogue)
- Renommer le script - pour changer le nom du groupe ou du script sélectionné
- Nouveau VBScript/JScript/ JavaScript - ajoute un nouveau script au groupe d'utilisateurs sélectionné
- Importer un script de flux de travail - pour afficher la boîte de dialogue « Navigateur » à travers laquelle vous recherchez et sélectionnez un fichier source de script de flux de travail (.vbs) à importer dans le dossier de script de flux de travail
- Supprimer le groupe/script - pour supprimer le groupe d'utilisateurs ou le script sélectionné







Vous pouvez également déplacer ou copier un script d'un dossier de scripts utilisateur à un autre ; à :










- Déplacez un script, mettez-le en surbrillance dans la fenêtre Scriptant et faites-le glisser dans le dossier des scripts utilisateur auquel il appartient maintenant
- Copiez un script, mettez-le en surbrillance dans la fenêtre Scriptant et appuyez sur Ctrl pendant que vous le faites glisser dans le dossier des scripts utilisateur dans lequel le dupliquer

Accéder

Ruban	Spécialiser > Outils > Script Bibliothèque
-------	--

Barre d'outils de script

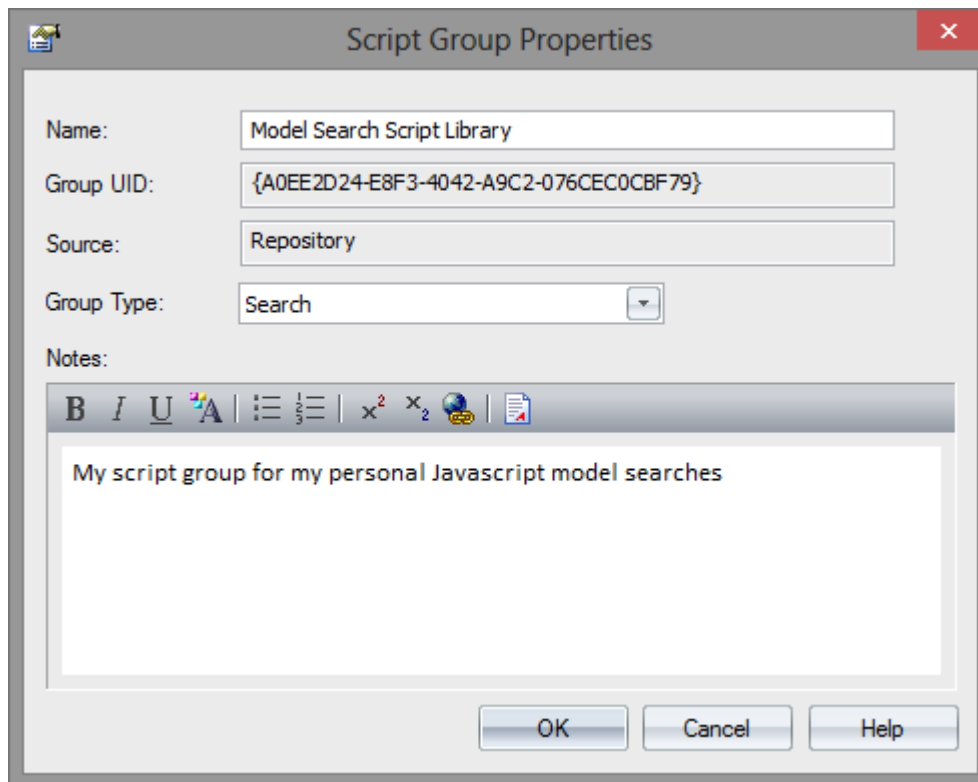
Icône	Action
	<p>Créez un nouveau groupe de scripts ; cette option affiche un petit menu des types de groupe de scripts que vous pouvez créer, à savoir :</p> <ul style="list-style-type: none"> • Groupe normal () • Groupe de fenêtres du  Navigateur • Groupe  Diagramme • Groupe de flux de travail () • Groupe de recherche () • Modèle de recherche de modèles <p>Le nouveau groupe est ajouté à la fin de la liste dans la fenêtre Scriptant , avec le texte « Nouveau groupe » en surbrillance afin que vous puissiez saisir le nom du</p>

	groupe.
	<p>Créez un nouveau fichier de script dans le groupe de scripts sélectionné ; cela affiche un petit menu des types de script que vous pouvez créer, à savoir :</p> <ul style="list-style-type: none"> • VBScript () • JScript () • JavaScript () <p>Le nouveau script est ajouté à la fin de la liste dans le groupe sélectionné, avec le texte « Nouveau script » en surbrillance afin que vous puissiez saisir le nom du script.</p>
	Actualisez l'arborescence de script dans la fenêtre Scriptant ; cette icône recharge également toutes les modifications apportées à un script de workflow.
	<p>Compilez et exécutez le script sélectionné.</p> <p>La sortie du script est écrite dans l'onglet "Script" de la fenêtre de sortie système, que vous affichez à l'aide du bouton Vue Script Output.</p>
	Arrêter un script en cours d'exécution ; l'icône est désactivée si aucun script n'est en cours d'exécution.
	<p>Supprimer un script du modèle ; vous ne pouvez pas utiliser cette icône pour supprimer un groupe de scripts (voir l'élément ' Menu Contexte ' précédent), les scripts du groupe ' Scripts locaux ' ou un script en cours d'exécution.</p> <p>Le système vous promps à confirmer la suppression uniquement si la case 'Confirmer la suppression' est cochée dans le panneau 'Project Navigateur ' de la page 'Général' de la boîte de dialogue 'Préférences' ; si cette option n'est pas sélectionnée, aucune prompt ne s'affiche.</p> <p>La suppression des scripts est permanente - les scripts ne peuvent pas être récupérés.</p>
	Affichez la fenêtre Sortie système avec les résultats du dernier script exécuté affiché dans l'onglet 'Script'.

Notes

- Cette facilité est disponible dans les éditions Corporate , Unified et Ultimate
- Si vous ajoutez, supprimez ou modifiez un script, vous devrez peut-être recharger le modèle pour que les modifications prennent effet
- Si vous choisissez de supprimer un groupe de scripts contenant des scripts, le système vous promps toujours à confirmer l'action, quels que soient les paramètres système pour les opérations de suppression ; soyez certain que vous avez l'intention de supprimer le groupe et ses scripts avant de confirmer la suppression - la suppression des groupes de scripts et des scripts est permanente

Propriétés du groupe de scripts









Lorsque vous créez un script, vous le développez au sein d'un groupe de scripts dont les propriétés déterminent comment ce script doit être mis à la disposition de l'utilisateur - via le menu contextuel de la fenêtre Navigateur pour opérer sur des objets d'un type spécifique, ou via un contexte de diagramme menu. Vous créez un groupe de scripts à l'aide de la première icône de la barre d'outils de la fenêtre Scriptant .

Accéder

Ruban	Spécialiser > Outils > Script Bibliothèque > Scripts > cliquez-droit sur [Nom du groupe] > Propriétés du groupe
-------	---

Définir les propriétés du groupe de Propriétés

Champ/Bouton	Action
Nom	Type le nom du groupe de scripts.
UID de groupe	(Lecture seule) GUID attribué automatiquement au groupe.
La source	(Lecture seule) L'emplacement du gabarit utilisé pour créer le script.
Type de groupe	Cliquez sur la flèche déroulante et sélectionnez le type de script contenu dans le groupe ; cela peut être l'un des suivants :

	<ul style="list-style-type: none"> • Normal - () Scripts de modèle généraux • Fenêtre Navigateur - () Scripts répertoriés et exécutables depuis l'option de menu contextuel ' Scripts ' de la fenêtre Navigateur • Workflow - () Scripts exécutés par le moteur de workflow d' Enterprise Architect ; vous ne pouvez créer que des scripts VB de ce type • Recherche - () Scripts pouvant être exécutés en tant que recherches de modèles ; ces scripts sont listés dans le champ 'Recherche' de la fenêtre Recherche de Modèle , dans la dernière catégorie de la liste • Diagramme - () Scripts exécutables depuis le sous-menu ' Scripts ' du menu contextuel du diagramme • Rechercher dans Projet - () Scripts pouvant être exécutés depuis le sous-menu ' Scripts ' d'un menu contextuel dans la vue Modèle Search, sur les résultats d'une recherche SQL exécutée avec succès qui inclut CLASSGUID et CLASSTYPE, ou une Query- recherche construite • Élément - Scripts pouvant être exécutés à partir du sous-menu « Scripts » des menus contextuels des éléments ; accessible depuis la fenêtre du Navigateur , les vues Diagramme , Recherche de Modèle , Liste d'éléments, Paquetage Navigateur et Gantt • Paquetage - Scripts exécutables depuis le sous-menu ' Scripts ' des menus contextuels de Paquetage ; accessible depuis la fenêtre du Navigateur • Diagramme - Scripts pouvant être exécutés à partir de l'option de menu contextuel « Scripts » pour les diagrammes ; accessible depuis la fenêtre Navigateur et les diagrammes • Lien - Scripts pouvant être exécutés à partir de l'option de menu contextuel « Scripts » pour les connecteurs ; accessible depuis les diagrammes
Notes	Type tous les commentaires dont vous avez besoin concernant ce groupe de scripts.

JavaScript Math Library

Le légendaire Cephes Math Bibliothèque est entièrement et étroitement intégré au moteur JavaScript disponible dans Enterprise Architect . Cette bibliothèque est une collection de plus de 400 routines mathématiques de haute qualité pour des applications scientifiques et d'ingénierie, offrant une vaste gamme de potentiel mathématique pour les modélisateurs souhaitant faire passer leurs modèles d'ingénierie et de systèmes au niveau supérieur.

La bibliothèque de fonctions implémente la norme double précision IEEE Std 754.

- [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](#)

Arithmétique et Algèbrique

- [sqrt](#) - racine carrée
- [lsqrt](#) - racine carrée integer
- [cbrt](#) - racine cubique
- [polevl](#), [p1evl](#) - évaluer le polynôme
- [chbevl](#) - évaluer la série Chebyshev
- [round](#) - round à la valeur integer la plus proche
- [ceil](#) - tronquer vers le haut en integer
- [floor](#) - tronquer vers le bas en integer
- [frexp](#) - extraire l'exposant
- [ldexp](#) - ajoute un integer à l'exposant
- [fabs](#) - valeur absolue
- [signbit](#) de signe - renvoie le bit de signe sous forme d' int
- [isnan](#) - test de nombre
- [isfinite](#) - test fini
- [poladd](#) - ajouter des polynômes
- [polsub](#) - soustraire des polynômes
- [polmul](#) - multiplier les polynômes
- [poldiv](#) - diviser des polynômes
- [polsbt](#) - variable polynomiale de substitution
- [poleva](#) - évaluer le polynôme
- [polclr](#) - met tous les coefficients à zéro
- [polmov](#) - copier les coefficients

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

Isqrt

Integer square root.

SYNOPSIS:

```
long x, y;  
long Isqrt();  
y = Isqrt(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 $\text{Isqrt}(2,147,483,647) = 46341$.

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 , p1evl

Evaluate polynomial.

SYNOPSIS:

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

DESCRIPTION:

Evaluates polynomial of degree N:

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

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 = [ 7.87311395793093628397E-4, -2.29549961613378126380E-4,
             -2.68132617805781232825E-3, 3.47222221605458667310E-3,
             8.3333333333482257126E-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

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

of Chebyshev polynomials T_i 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 $x \rightarrow 2(2x - b - a)/(b-a)$ 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 $x \rightarrow 2(2ab/x - b - a)/(b-a)$. If b is infinity, this becomes $x \rightarrow 4a/x - 1$.

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 $1/(2*\text{MACHEP})$, 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,

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

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:

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

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,

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

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,

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

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,

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

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,

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

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,

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

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,

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

Exponentiel et Trigonométrie

- [acos](#) - Arc cosine
- [acosh](#) - Arc cosine hyperbolique
- [asinh](#) - Arc sine hyperbolique
- [atanh](#) - Arc tangente hyperbolique
- [asin](#) - Arcsinus
- [atan](#) - Arctangente
- [atan2](#) - Arctangente correcte du quadrant
- [cos](#) - Cosinus
- [cosdg](#) - Cosinus de arg en degrés
- [exp](#) - Exponentiel , base e
- [exp2](#) - Exponentiel , base 2
- [exp10](#) - Exponentiel , base 10
- [cosh](#) - cosine hyperbolique
- [sinh](#) - sine hyperbolique
- [tanh](#) - Tangente hyperbolique
- [log](#) - Logarithme, base e
- [log2](#) - Logarithme, base 2
- [log10](#) - Logarithme, base 10
- [pow](#) - Pouvoir
- [powi](#) - Puissance Integer
- [sin](#) - Sine
- [sindg](#) - Sine d'arg en degrés
- [tan](#) - Tangente
- [tandg](#) - Tangente de arg en degrés

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

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 $1 \leq x < 1.5$, a rational approximation:

$$\sqrt{z} * P(z)/Q(z)$$

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

$$\operatorname{acosh}(x) = \log(x + \sqrt{(x-1)(x+1)}).$$

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, $\text{asinh}(x) = \log(x + \text{sqrt}(1 + x*x))$.

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:

$$\operatorname{atanh}(x) = 0.5 * \log((1+x)/(1-x)).$$

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:

$$\text{asin}(x) = \pi/2 - 2 \text{asin}(\sqrt{(1-x)/2}).$$

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 $-\pi < z \leq +\pi$, args (y,x);

else ANSIC = 0 for range 0 to 2π , 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:

$$x = k + f$$

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

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:

$$\exp(X(1+\delta)) = \exp(X) (1 + X*\delta + \dots)$$

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:

$$x = k + f$$

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

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	$x < -\text{MAXL2}$	0.0
overflow	$x > \text{MAXL2}$	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 $|f| < 0.5 \log_{10}(2)$.

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	$x < -\text{MAXL10}$	0.0
overflow	$x > \text{MAXL10}$	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 $|x| \leq 1$, 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

Un algorithme naturel.

SYNOPSIS:

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

LA DESCRIPTION:

Renvoie la base e (2,718...) logarithme de x.

L'argument est séparé en ses parties exposant et fractionnaire. Si l'exposant est compris entre -1 et +1, le logarithme de la fraction est approximé par :

$$\log (1+x) = x - 0,5 x^{**2} + x^{**3} P(x)/Q(x).$$

Sinon, en posant $z = 2(x-1)/(x+1)$,

$$\log (x) = z + z^{**3} P(z)/Q(z).$$

PRÉCISION:

Erreur relative:

domaine arithmétique nombre d'essais crête 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

Dans les tests sur l'intervalle [+MAXNUM], les logarithmes des arguments aléatoires étaient uniformément répartis sur [0,MAXLOG].

MESSAGES D'ERREUR:

singularité : $x = 0$; renvoie -INFINI

domaine : $x < 0$; renvoie 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	$x^{**}y > \text{MAXNUM}$	INFINITY
underflow	$x^{**}y < 1/\text{MAXNUM}$	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 n th power. The routine efficiently decomposes n as a sum of powers of two. The desired power is a product of two-to-the- k th 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

Intégrale Exponentielle

- [expn](#) - Intégrale Exponentielle
- [shichi](#) - Intégrales sine et cosine hyperboliques
- [sici](#) - Intégrales Sine et cosine_

expn

Exponential integral En.

SYNOPSIS:

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

DESCRIPTION:

Evaluates the exponential integral.

$$E(x) = \int_0^{\infty} \frac{e^{-xt}}{1+nt} dt.$$

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

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

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

where $\text{eul} = 0.57721566490153286061$ 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 $|\text{Chi}| > 1$:

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:

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

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

where $\text{eul} = 0.57721566490153286061$ 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](#) - log naturel de beta
- [fac](#) - factoriel
- [gamma](#) - gamma
- [lgam](#) - logarithme de la fonction gamma
- [incbet](#) - intégrale beta incomplète
- [incbi](#) - inverse de l'intégrale beta incomplète
- [igam](#) - intégrale gamma incomplète
- [igamc](#) - intégrale gamma complétée
- [igami](#) - intégrale gamma inverse
- [psi](#) - Fonction Psi (digamma)
- [rgamma](#) - Gamma réciproque

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	<code>log(beta) > MAXLOG</code>	0.0
	<code>a or b < 0 integer</code>	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 $i = 1 * 2 * 3 * \dots * i$.

$\text{fac}(0) = 1.0$.

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 $i > 55$, $\text{fac}(i) = \text{gamma}(i+1)$;

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 $|x| \leq 34$ 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:

$$\frac{\int_0^x t^{a-1} (1-t)^{b-1} dt}{\int_0^1 t^{a-1} (1-t)^{b-1} dt}$$

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

$$1 - \text{incbet}(a, b, x) = \text{incbet}(b, a, 1-x).$$

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.

arithmetic	domain	# trials	Relative error	
			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:

$$\text{incbet}(a, b, x) = y .$$

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

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

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

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

$$\text{igamc}(a,x) = 1 - \text{igam}(a,x)$$

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

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:

Tested at random a, x .

	a	x	Relative error:		
			domain	# trials	peak
IEEE arithmetic	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

$$\text{igamc}(a, x) = p.$$

Starting with the approximate value

$$x = a t^3$$

where

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

and

$$d = 1/9a,$$

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

ACCURACY:

Tested at random a, p in the intervals indicated.

arithmetic	a p		Relative error:		
	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:

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

is the logarithmic derivative of the gamma function.

For integer x:

$$\psi(n) = -\gamma + \sum_{k=1}^{n-1} \frac{1}{k}$$

This formula is used for $0 < n \leq 10$. 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:

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

where the B_{2k} 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.

Fonction d'erreur

- [erf](#) - Fonction d'erreur
- [erfc](#) - Fonction d'erreur complétée
- [dawsn](#) - Intégrale de Dawson
- [fresnl](#) - Intégrale de Fresnel

erf

Error function.

SYNOPSIS:

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

DESCRIPTION:

The integral is

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

The magnitude of x is limited to 9.231948545 for DEC arithmetic; 1 or -1 is returned outside this range.

For $0 \leq |x| < 1$, $\text{erf}(x) = x * P4(x**2)/Q5(x**2)$; otherwise $\text{erf}(x) = 1 - \text{erfc}(x)$.

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:

$$1 - \operatorname{erf}(x) = \frac{\operatorname{erfc}(x)}{2}$$

$$\operatorname{erfc}(x) = \frac{2}{\sqrt{\pi}} \int_x^{\infty} \exp(-t^2) dt$$

For small x , $\operatorname{erfc}(x) = 1 - \operatorname{erf}(x)$; otherwise rational approximations are computed.

A special function `exp2.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

dawson

Dawson's Integral.

SYNOPSIS:

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

DESCRIPTION:

Approximates the integral

$$\text{dawson}(x) = \frac{\exp(-x^2)}{\sqrt{\pi}} \int_0^x \exp(t^2) dt$$

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

$$C(x) = \int_0^x \cos(\pi/2 t^2) dt,$$

$$S(x) = \int_0^x \sin(\pi/2 t^2) dt.$$

The integrals are evaluated by a power series for $x < 1$. For $x \geq 1$ 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  
}
```

Bessel

- [airy](#) - Fonction aéré
- [j0](#) - Bessel , ordre 0
- [j1](#) - Bessel , ordre 1
- [jn](#) - Bessel , ordre n
- [jv](#) - Bessel , ordre non entier
- [y0](#) - Bessel , seconde espèce, ordre 0
- [y1](#) - Bessel , seconde espèce, ordre 1
- [yn](#) - Bessel , seconde espèce, ordre n
- [yv](#) - Bessel , ordre non entier
- [i0](#) - Bessel modifié, ordre 0
- [i0e](#) - mise à l'échelle exponentielle i0
- [i1](#) - Bessel modifié , ordre 1
- [i1e](#) - i1 mis à l'échelle exponentiellement
- [iv](#) - Bessel modifié, non int. ordre
- [k0](#) - Bessel modifié, 3e type, ordre 0
- [k0e](#) - k0 mis à l'échelle exponentiellement
- [k1](#) - Bessel modifié, 3e type, ordre 1
- [k1e](#) - k1 mis à l'échelle exponentiellement
- [kn](#) - Bessel modifié, 3e type, ordre 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:

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

1 2 3 8

2

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 $(8, \text{infinity})$. 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 $j_n(x)$ to $j_0(x)$ is computed by backward recurrence. First the ratio j_n/j_{n-1} is found by a continued fraction expansion. Then the recurrence relating successive orders is applied until j_0 or j_1 is reached.

If $n = 0$ or 1 the routine for j_0 or j_1 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:

$$y_0(x) = R(x) + 2 * \log(x) * j_0(x) / \text{PI}.$$

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 $y_0(x) < 1$; 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 $(8, \text{infinity})$. 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 $i_0(x) = j_0(ix)$.

The range is partitioned into the two intervals $[0,8]$ and $(8, \text{infinity})$. 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 $(8, \text{infinity})$. 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 $I_v(x) = J_v(ix)$. Here, it is computed in terms of the confluent hypergeometric function, according to the formula:

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

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	$x \leq 0$	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.

Hypergéométrie

- [hyperg](#) - hypergéométrie confluent
- [hyp2f1](#) - Fonction hypergéométrique de Gauss
- [hyp2f0](#) - 2F0
- [onef2](#) - 1F2
- [threef0](#) - 3F0

hyperg

Confluent hypergeometric function.

SYNOPSIS:

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

DESCRIPTION:

Computes the confluent hypergeometric function

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

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 $0 \geq a > b$.

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

lstd:

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 ${}_2F_1$.

SYNOPSIS:

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

DESCRIPTION:

$$\text{hyp2f1}(a, b, c, x) = F(a, b; c; x)$$

$$= 1 + \sum_{k=0}^{\infty} \frac{a(a+1)\dots(a+k) b(b+1)\dots(b+k)}{c(c+1)\dots(c+k) (k+1)!} x^k$$

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 10^{-12} .

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

hyp2f0

See the [hyperg](#) Help topic.

onef2

Voir la rubrique d'aide de la [struve](#) .

threef0

Voir la rubrique d'aide de la [struve](#) .

Elliptique

- [elpe](#) - intégrale elliptique complète (E)
- [ellie](#) - intégrale elliptique incomplète (E)
- [ellpk](#) - intégrale elliptique complète (K)
- [ellik](#) - intégrale elliptique incomplète (K)
- [ellpj](#) - Fonctions elliptiques jacobiniennes

ellpe

Complete elliptic integral of the second kind.

SYNOPSIS:

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

DESCRIPTION:

Approximates the integral

$$E(m) = \int_0^{\pi/2} \sqrt{1 - m \sin^2 t} \, dt$$

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:

$$E(\phi|m) = \int_0^{\phi} \sqrt{1 - m \sin^2 t} dt$$

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

ACCURACY:

Tested at random arguments with ϕ 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:

$$K(m) = \int_0^{\pi/2} \frac{dt}{\sqrt{1 - m \sin^2 t}}$$

where $m = 1 - m_1$, using the approximation:

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

The argument m_1 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:

$$F(\text{phi_}m) = \int_0^{\text{phi}} \frac{dt}{\sqrt{1 - m \sin^2 t}}$$

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 $\text{sn}(u|m)$, $\text{cn}(u|m)$, and $\text{dn}(u|m)$ 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 $\text{ellpk}(1.0-m)$.

Relation to incomplete elliptic integral:

If $u = \text{ellik}(\text{phi}, m)$, then $\text{sn}(u|m) = \sin(\text{phi})$, and $\text{cn}(u|m) = \cos(\text{phi})$.

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 $\text{phi} < \pi/2$.

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 $\text{sn}(u+v)$ was $4e-16$ (absolute). Also tested by the earlier relation to the incomplete elliptic integral. Accuracy deteriorates when u is large.

Probabilité

- [bdtr](#) - Distribution binomiale
- [bdtrc](#) - Binôme complémenté
- [bdtri](#) - Binôme inverse
- [chdtr](#) - Distribution du chi carré
- [chdtrc](#) - Chi carré complémenté
- [chdtri](#) - Chi carré inverse
- [fdtr](#) - distribution F
- [fdtrc](#) - F complété
- [fdtri](#) - Distribution F inverse
- [gdtr](#) - Distribution gamma
- [gdtrc](#) - gamma complémenté
- [nbdtr](#) - Distribution binomiale négative
- [nbdtrc](#) - Binôme négatif complété
- [ndtr](#) - Distribution normale
- [ndtri](#) - Distribution normale inverse
- [pdtr](#) - Loi de Poisson
- [pdtrc](#) - Poisson complémenté
- [pdtri](#) - Distribution de Poisson inverse
- [stdtr](#) - Distribution t de Student

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:

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

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

$$y = \text{bdtr}(k, n, p) = \text{incbet}(n-k, k+1, 1-p).$$

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:			
	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:

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

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

$$y = \text{bdtrc}(k, n, p) = \text{incbet}(k+1, n-k, p).$$

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 = bdtri(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:

$$1 - p = \text{incbi}(n-k, k+1, y).$$

ACCURACY:

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

a,b	domain	# trials	Relative error:	
			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	$k < 0, n \leq k$	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.

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

where x is the Chi-square variable.

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

$$y = \text{chdtr}(v, x) = \text{igam}(v/2.0, x/2.0).$$

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:

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

where x is the Chi-square variable.

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

$$y = \text{chdtr}(v, x) = \text{igamc}(v/2.0, x/2.0).$$

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:

$$x/2 = \text{igami}(df/2, y);$$

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

$$P(x) = \text{incbet}(df1/2, df2/2, (df1*x)/(df2 + df1*x)).$$

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:		
	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).

$$1-P(x) = \frac{\int_x^{\infty} t^{a-1} (1-t)^{b-1} dt}{B(a,b)}$$

The incomplete beta integral is used, according to the formula

$$P(x) = \text{incbet}(df2/2, df1/2, (df2/(df2 + df1*x))).$$

ACCURACY:

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

x	a,b		Relative error:		
	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:

$$z = \text{incbi}(\text{df2}/2, \text{df1}/2, p)$$

$$x = \text{df2} (1-z) / (\text{df1} z).$$

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

$$z = \text{incbi}(\text{df1}/2, \text{df2}/2, p)$$

$$x = \text{df2} z / (\text{df1} (1-z)).$$

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 $p \leq 0$ or $p > 1$ 0.0
 $v < 1$

gdr

Gamma distribution function.

SYNOPSIS:

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

DESCRIPTION:

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

$$y = \frac{1}{\Gamma(b)} \int_0^x t^{b-1} e^{-at} dt$$

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:

$$y = \int_x^{\infty} \frac{b^{-a}}{\Gamma(a)} t^{a-1} e^{-bt} dt$$

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:

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

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:

$$y = \text{nbdtr}(k, n, p) = \text{incbet}(n, k+1, p).$$

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:

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

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

$$y = \text{nbdtrc}(k, n, p) = \text{incbet}(k+1, n, 1-p).$$

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:

$$\text{ndtr}(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^x \exp(-t/2) dt$$

$$= (1 + \text{erf}(z)) / 2$$

$$= \text{erfc}(z) / 2$$

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 $x/\sqrt{2\pi} = w + w**3 R(w**2)/S(w**2)$.

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	$x \leq 0$	-MAXNUM
domain	$x \geq 1$	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:

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

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

$$y = \text{pdtr}(k, m) = \text{igamc}(k+1, m).$$

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:

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

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

$$y = \text{pdtrc}(k, m) = \text{igam}(k+1, m).$$

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

$$m = \text{igami}(k+1, y).$$

ACCURACY:

See the [igami](#) Help topic.

ERROR MESSAGES:

message	condition	value returned
domain	$y < 0$ or $y \geq 1$	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:

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

Relation to incomplete beta integral:

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

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 $1 \leq k \leq 25$. 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

Divers

- [polylog](#) - Polylogarithmes
- [spence](#) - Dilogarithme
- [zetac](#) - Fonction Zeta de Riemann
- [zeta](#) - Fonction zeta à deux arguments
- [struve](#) - Fonction Struve

polylog

Polylogarithms.

SYNOPSIS:

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

The polylogarithm of order n is defined by the series:

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

For $x = 1$,

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

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

$$Li(x) = \int_0^x \frac{-\ln(1-t)}{t} dt = \int_x^{1-x} \frac{\ln t}{1-t} dt = \text{spence}(1-x).$$

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:

$$\text{spence}(x) = - \int_0^x \frac{\log t}{t-1} dt$$

for $x \geq 0$. 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:

inf.
 - -x

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

Is related to the Riemann zeta function by:

$$\text{Riemann zeta}(x) = \zeta(x) + 1.$$

Extension of the function definition for $x < 1$ is implemented.

Zero is returned for $x > \log_2(\text{MAXNUM})$.

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:

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

where $x > 1$ and q is not a negative integer or zero.

The Euler-Maclaurin summation formula is used to obtain the expansion

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

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

where the B_{2j} are Bernoulli numbers.

Note that $\text{zeta}(x, 1) = \text{zeta}(x) + 1$.

(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 $H_v(x)$ 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 $Y_v(x)$ with noninteger v .

ACCURACY:

Not accurately characterized, but spot checked against tables.

Matrice

- [fftr](#) - Transformée de Fourier rapide
- [simq](#) - Équations linéaires simultanées
- [minv](#) - Inversion de Matrice
- [mmmpy](#) - Multiplication Matrice
- [mvmpy](#) - Matrice de temps de matrice
- [mtransp](#) - Transposition Matrice
- [eigens](#) - vecteurs propres (matrice symétrique)_

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 $\sin(2 \text{ PI } k / n)$.

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

DESCRIPTION:

B, X, IPS are vectors of length n.

A is an $n \times n$ 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],
  ]
}
```

```
[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, $A[\text{row}, \text{column}] = A[(\text{row} * \text{row} + \text{row}) / 2 + \text{column}]$ 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;  
}
```

Intégration Numérique

- [simpsn](#) - règle de Simpson

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.

Arithmétique Complexe

- [cadd](#) - Addition complexe
- [csub](#) - Soustraction
- [cmul](#) - Multiplication
- [cdiv](#) - Division
- [cabs](#) - valeur absolue
- [csqrt](#) - Racine carrée

cadd

Addition.

SYNOPSIS:

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

```
cplx *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
}cplx;
```

```
cplx *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

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

$$\operatorname{Re} w = y / 2 \operatorname{Im} w.$$

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  
}
```

Exponentiel Complexe et Trigonométrie

- [cexp](#) - Exponentiel
- [clog](#) - Logarithme
- [ccos](#) - Cosinus
- [cacos](#) - Arc cosine
- [csin](#) - Sine
- [casin](#) - Arc sine
- [ctan](#) - Tangente
- [catan](#) - Arc tangente
- [ccot](#) - Cotangente

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:

$w = \arccos z = \text{PI}/2 - \arcsin z$.

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:

$$2$$
$$w = -i \operatorname{clog}(iz + \operatorname{csqrt}(1 - z^2)).$$

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 $\operatorname{csin}(\operatorname{casin}(z)) = z$.

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 $\text{ctan} * \text{ccot} = 1$ and $\text{catan}(\text{ctan}(z)) = z$.

catan

Complex circular arc tangent.

SYNOPSIS:

```
void catan();
```

```
cmplx z, w;
```

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

DESCRIPTION:

If

$$z = x + iy,$$

then

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

Re w = - arctan(-----) + k PI

$$\frac{2}{(1-x^2-y^2)}$$

$$(1-x^2-y^2)$$

$$\frac{(x^2+y^2)}{1+(x^2+y^2)}$$

$$\frac{1}{(x^2+(y+1)^2)}$$

Im w = - log(-----)

$$\frac{4}{(x^2+y^2)}$$

$$(x^2+(y-1)^2)$$

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 $\operatorname{catan}(\operatorname{catan}(z)) = z$, 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.

les erreurs

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" ];

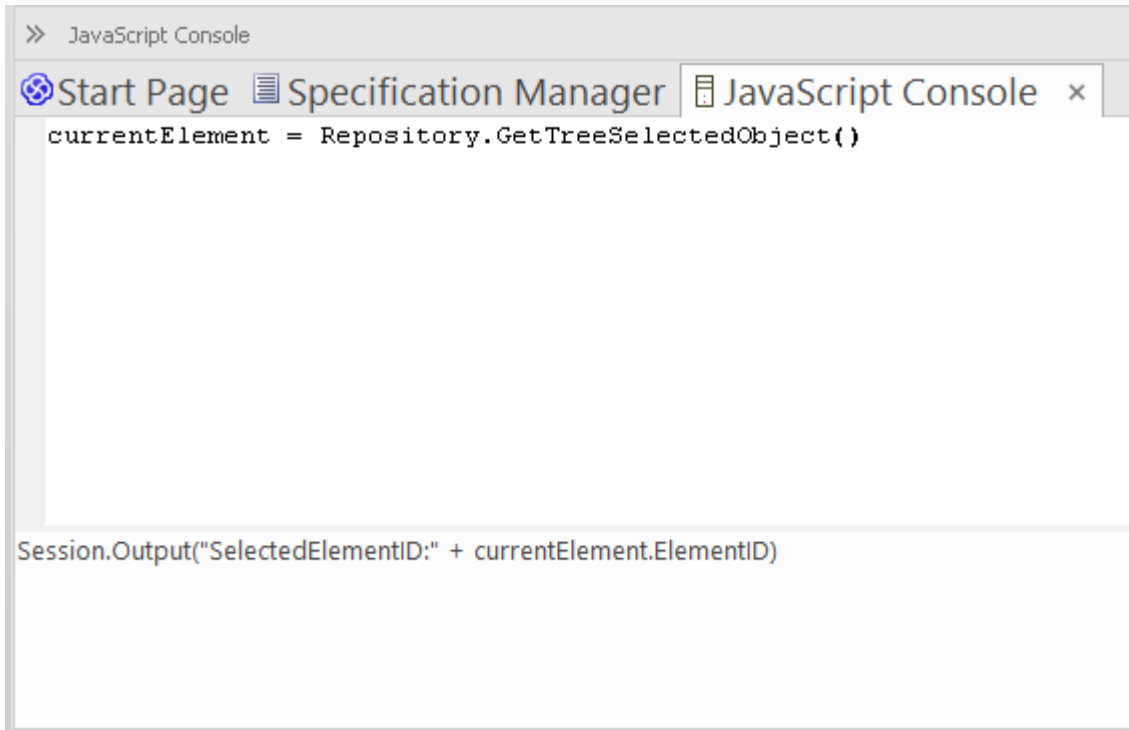
function printError()
{
    var er = cephes.geterror();
    if(er>0)
    {
        Session.Output( "cephas error " + er + " " + cephes_errors[er]);
    }
}
```

Testing for error

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

JavaScript Console

La console JavaScript est un interpréteur de ligne de commande qui accepte les commandes JavaScript sur une seule ligne qui seront exécutées une par une. Vous tapez les commandes dans le panneau de saisie de texte au bas de l'écran et, lorsque vous appuyez sur la touche Entrée pour exécuter la commande, elle est ajoutée à la fenêtre de sortie supérieure avec toute sortie de la commande. Considérez cet exemple.



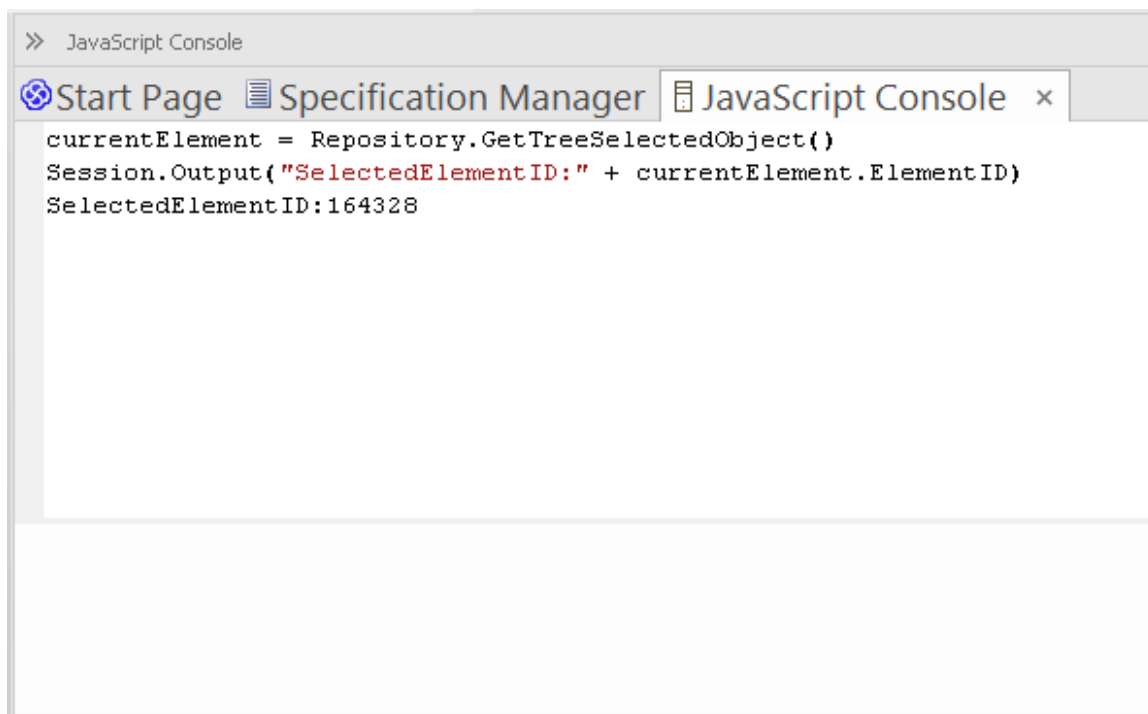
Dans le panneau du bas, l'utilisateur a saisi :

```
currentElement = Référentiel .GetTreeSelectedObject()
```

Ils ont appuyé sur la touche Entrée et la commande s'est affichée dans le panneau supérieur. L'utilisateur a alors tapé, dans le panneau inférieur :

```
Session.Output("ID élément sélectionné :"+élémentcourant.IDélément)
```

Lorsqu'ils appuient sur la touche Entrée, la console affiche cette commande et la sortie de la commande.

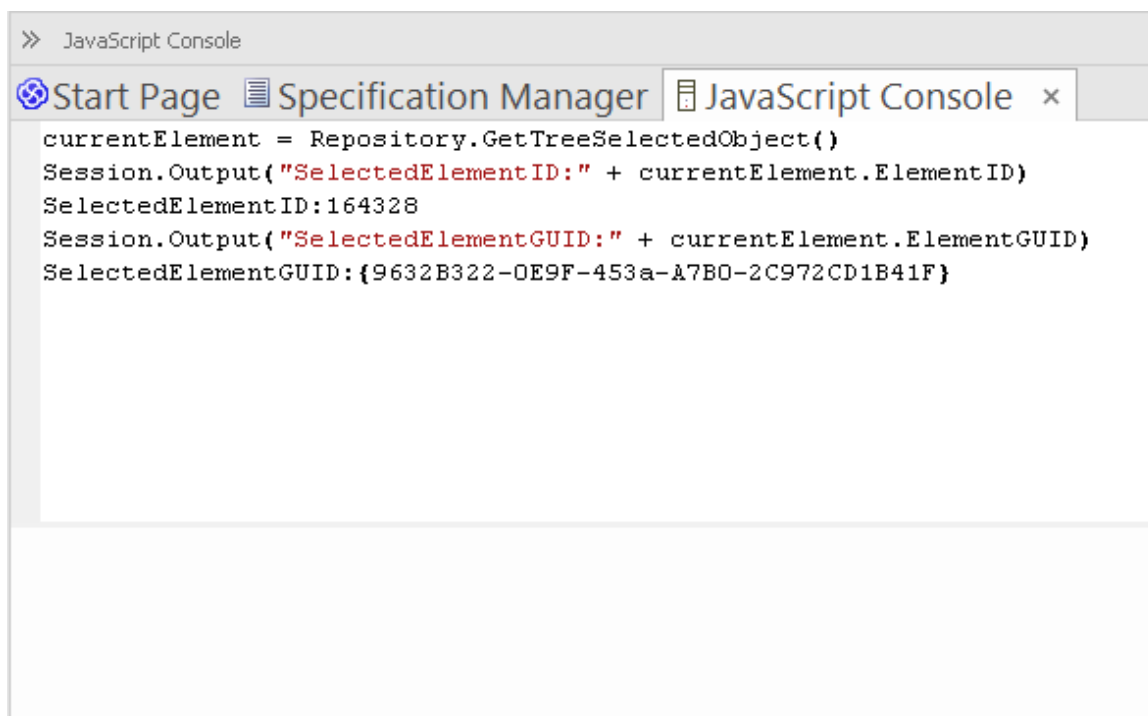


```
>> JavaScript Console
Start Page Specification Manager JavaScript Console x
currentElement = Repository.GetTreeSelectedObject()
Session.Output("SelectedElementID:" + currentElement.ElementID)
SelectedElementID:164328
```

L'utilisateur saisit alors une troisième commande :

```
Session.Output("SelectedElementGUID : " + currentElement.ElementGUID )
```

Cela se traduit par la sortie affichée ici, dans le panneau supérieur :



```
>> JavaScript Console
Start Page Specification Manager JavaScript Console x
currentElement = Repository.GetTreeSelectedObject()
Session.Output("SelectedElementID:" + currentElement.ElementID)
SelectedElementID:164328
Session.Output("SelectedElementGUID:" + currentElement.ElementGUID)
SelectedElementGUID:{9632B322-OE9F-453a-A7B0-2C972CD1B41F}
```

Cette fonctionnalité est disponible dans les éditions Corporate , Unified et Ultimate d' Enterprise Architect .

Accéder

Ruban	Spécialiser > Outils > JavaScript
-------	-----------------------------------

Simuler > Console > JavaScript

Commandes de la console

Les commandes de la console sont précédées du signe ! personnage et demandez à la console d'effectuer une action.

Les commandes de console disponibles incluent :

- !clear - efface l'affichage de la console
- !save - enregistre l'affichage de la console dans un fichier
- !help - affiche une liste de commandes
- !close - ferme la console
- !include <scriptname> - exécute l'élément de script nommé ; scriptname est au format GroupName.ScriptName (les espaces sont autorisés dans les noms)
- ? - répertorie les commandes (comme !help)
- ? <nom de la variable ou de la fonction> - renvoie la valeur .

Pour lister ces commandes sur l'onglet 'Console' lui-même, tapez ? dans le panneau inférieur (sans le caractère ! précédent) et appuyez sur la touche Entrée.

Si vous avez l'intention d'exécuter des scripts, vous souhaitez peut-être Scriptant Bibliothèque , afin que vous puissiez voir les scripts disponibles à appeler. Sélectionnez l'option de ruban "Spécialiser > Outils > Bibliothèque de scripts".

Fenêtre de la console

La fenêtre de la console est un interpréteur de ligne de commande grâce auquel vous pouvez rapidement activer un moteur de script et entrer des commandes pour agir sur le script (JScript, JavaScript et VBScript).

Vous pouvez ouvrir la fenêtre de la console pour JavaScript via les rubans Simuler et Spécialiser. Vous pouvez ouvrir la fenêtre de la console pour VBScript et JScript via le ruban Spécialiser.

Pour les trois langages de script, vous tapez les commandes dans le champ en bas de la fenêtre ; lorsque vous appuyez sur la touche Entrée, la console de script exécute les commandes et affiche immédiatement toute sortie. Les commandes de la console sont décrites dans la rubrique d'aide de la *JavaScript Console* .

Accéder

Ruban	Spécialiser > Outils > JavaScript Spécialiser > Outils > VBScript Spécialiser > Outils > JScript Simuler > Console > JavaScript
-------	--

Notes

- Cette facilité est disponible dans les éditions Corporate , Unified et Ultimate

Interface Solveurs

L'interface Solveurs vous permet d'invoquer un ensemble de commandes en JavaScript qui définissent et exécutent une classe Solveur pour effectuer des opérations mathématiques sur les données. La fonction principale de la classe Solveur est de fournir une intégration avec des outils externes tels que MATLAB et Octave lors d'une simulation, et soit d'exposer les résultats dans Octave ou MATLAB, soit de les ramener dans Enterprise Architect pour y être représentés, peut-être dans un graphique dynamique. Plus généralement, l'interface Solveurs peut être utilisée dans des Add-Ins basés sur des modèles et des scripts personnalisés.

Pour appeler des fonctions depuis Octave ou MATLAB, vous devez vous familiariser avec les fonctions disponibles dans la bibliothèque de produit appropriée, comme décrit dans la documentation du produit.

Constructeur Solveur

Constructeur	La description
Solver(string solverName)	Crée un nouveau Solveur connecté à une nouvelle instance de l'application d'assistance spécifiée.

Solveur Méthodes

Méthode	La description
get(string name)	Récupère une valeur nommée dans l'environnement Solveur .
set(string name, object value)	Attribue une nouvelle valeur à une variable nommée dans l'environnement du Solveur .
exec(string name, string arguments, int returnValues)	Exécute une fonction nommée. Les fonctions réelles dépendront du type de Solveur utilisé.

Éditeur de Script

À l'aide de l' Éditeur de Script , vous pouvez effectuer un certain nombre d'opérations sur un fichier de script ouvert, telles que :

- Enregistrer les modifications dans le script actuel
- Enregistrer le script actuel sous un nom différent
- Exécuter le script
- Déboguer le script
- Arrêter le script en cours d'exécution
- Vue la sortie du script dans l'onglet ' Scripts ' de la fenêtre Sortie système

L'éditeur est basé sur, et offre les facilités de, l' Éditeur de Code commun dans la zone de travail de l'application.

Accéder

Ruban	<p>Spécialisez > Outils > Bibliothèque de scripts > développez le groupe de scripts et cliquez-droit sur [nom du script] > Modifier le script ou</p> <p>Spécialiser > Outils > Bibliothèque de scripts > développez le groupe de scripts et double-cliquez sur [nom du script]</p>
-------	---

Facilités

Facilité	Détail
Objets Scriptant	<p>Enterprise Architect ajoute aux fonctionnalités et fonctionnalités disponibles du langage de script de l'éditeur en fournissant des objets intégrés ; il s'agit soit de bibliothèques de Type fournissant Intelli-sense à des fins d'édition, soit d'objets d'exécution permettant d'accéder aux objets des types décrits dans les bibliothèques de Type .</p> <p>Les objets de script Intelli-sense disponibles sont :</p> <ul style="list-style-type: none"> • EA • MathLib • Système <p>Les objets de script d'exécution sont :</p> <ul style="list-style-type: none"> • Référentiel (Type : IDualRepository, une instance d'EA. Référentiel , l' Enterprise Architect Automation Interface) • Maths (Type : IMath, une instance de MathLib ; cela expose les fonctions de la bibliothèque mathématique Cephés à utiliser dans les scripts) • Session (Type : ISession, une instance de System)
Édition de script Intelli-sense (syntaxe requis)	<p>Intelli-sense est disponible non seulement dans 'l' Éditeur de Script ', mais aussi dans la 'Script Console' ; Intelli-sense dans sa forme la plus élémentaire est présenté pour la fonctionnalité intégrée du moteur de script.</p> <p>Pour Intelli-sense sur les objets de script Enterprise Architect supplémentaires (comme indiqué), vous devez déclarer des variables selon la syntaxe qui spécifie un</p>

	<p>type ; il n'est pas nécessaire d'utiliser cette syntaxe pour exécuter correctement un script, elle n'est présente que pour que le bon Intelli-sense puisse être affiché pour un élément.</p> <p>La syntaxe peut être vue dans, par exemple :</p> <pre>Dim e as EA.Element</pre> <p>Ensuite, lorsque vous tapez, dans ce cas, e., l'éditeur affiche une liste de fonctions membres et de propriétés de type e.</p> <p>Vous en sélectionnez un pour compléter la ligne de script ; vous pourriez donc taper :</p> <pre>VBTrace (e.</pre> <p>Au fur et à mesure que vous tapez la période, l'éditeur présente la liste appropriée et vous pouvez double-cliquer sur, par exemple, Résumé ; ceci est inséré dans la ligne, et vous continuez à taper ou à sélectionner le reste de l'instruction, dans ce cas en ajoutant l'espace de fin et la parenthèse :</p> <pre>VBTrace(e.Abstract)</pre>
Frappes	<p>Dans l' Éditeur de Script ou la Console, Intelli-sense est présenté sur ces frappes.</p> <ul style="list-style-type: none"> • Presse . (point) après un élément pour répertorier tous les membres pour le type de cet élément • Appuyez sur Ctrl + Espace sur un mot pour répertorier tous les éléments Intelli-sense dont le nom commence par la string au point où les touches ont été enfoncées • Appuyez sur Ctrl + Espace lorsque vous n'êtes pas sur un mot pour afficher tous les éléments Intelli-sense de niveau supérieur disponibles - ce sont les objets Intelli-sense déjà décrits plus toutes les méthodes et propriétés intégrées du langage de script actuel
Inclure les bibliothèques de scripts	<p>Une instruction <i>Inclure</i> (!INC) permet à un script de référencer des constantes, des fonctions et des variables définies par un autre script accessible dans la Scriptant de script. Les instructions Include sont généralement utilisées au début d'un script.</p> <p>Pour inclure une bibliothèque de scripts, utilisez cette syntaxe :</p> <pre>!INC [Nom du groupe de scripts].[Nom du script]</pre> <p>Par exemple:</p> <pre>!INC Scripts locaux .EAConstants-VBScript</pre>
Utilisation des fonctions mathématiques intégrées	<p>Diverses fonctions mathématiques sont disponibles dans l' Éditeur de Script , grâce à l'utilisation de l' objet intégré Maths.</p> <p>Vous pouvez accéder à l' objet Maths dans l' Éditeur de Script en tapant 'Maths' suivi d'un point. La fonctionnalité Intelli-sense affiche une liste des fonctions mathématiques disponibles fournies par la Bibliothèque Cephés Mathématiques .</p> <p>Par exemple:</p> <pre>Session.Output "La racine carrée de 9 est " & Maths. sqrt (9)</pre> <pre>Session.Sortie "2^10 = " & Maths. pow (2,10)</pre> <p>L' objet Maths est disponible dans les éditions Unified et Ultimate d' Enterprise Architect .</p>
Utilisation d'objets COM/ActiveX	<p>VBScript, JScript et JavaScript peuvent chacun créer et travailler avec des objets ActiveX / COM. Cela peut vous aider à travailler avec des bibliothèques externes ou à interagir avec d'autres applications externes à Enterprise Architect . Par exemple, la classe Scripting.FileSystemObject peut être utilisée pour lire et écrire des fichiers sur la machine locale. La syntaxe de création d'un nouvel objet varie légèrement pour chaque langage, comme illustré par ces exemples :</p>

	<p>VBScript :</p> <pre>set fsObject = CreateObject(" Scripting.FileSystemObject ")</pre> <p>JScript :</p> <pre>fsObject = new ActiveXObject(" Scripting.FileSystemObject ");</pre> <p>JavaScript :</p> <pre>fsObject = new COMObject(" Scripting.FileSystemObject ");</pre>
Utilisation de JavaScript avec des serveurs COM hors processus	<p>Les utilisateurs de JavaScript dans Enterprise Architect peuvent accéder à des serveurs COM hors processus. L'application doit être enregistrée sur la machine comme fournissant la prise en support du serveur local. La syntaxe pour créer ou obtenir une référence à un serveur hors processus est :</p> <pre>var serveur = new COMObject(progID , true);</pre> <p>où <i>progID</i> est l' ID de programme enregistré pour le composant COM (" Excel .Application", par exemple).</p>
Bibliothèque de scripts système	<p>Lorsqu'Enterprise Architect est installé sur votre système, il inclut une bibliothèque de scripts par défaut qui fournit un certain nombre de fonctions de script utiles, allant de simples fonctions de string à des fonctions permettant de définir votre propre importation et exportation CSV ou XML.</p> <p>Pour utiliser la bibliothèque de scripts, vous devez l'activer dans la boîte de dialogue 'MDG Technologies ' (option de ruban 'Spécialiser > Technologies > Gérer la technologie').</p> <p>Faites défiler la liste des technologies et cochez la case "Activé" contre "EAScriptLib".</p>

Notes

- L' Éditeur de Script est disponible dans les éditions Corporate , Unified et Ultimate
- Les scripts Enterprise Architect prennent en supporte la déclaration de variables pour correspondre aux types Enterprise Architect ; cela permet à l'éditeur de présenter Intelli-sense, mais n'est pas nécessaire à l'exécution du script

Objet de Session

L' objet d'exécution Session fournit un mécanisme d'entrée/rétroaction commun à tous les langages de script, donnant accès aux objets des types décrits dans la bibliothèque de Type système. Il est disponible via la fenêtre Scriptant pour tout script exécuter dans Enterprise Architect .

Propriétés

Propriétés	Détail
Attributes	<ul style="list-style-type: none"> • UserName - Renvoie le nom d'utilisateur Windows actuel (lecture seule) • Version - Renvoie la version de cet objet (lecture seule)
Méthodes	<ul style="list-style-type: none"> • Input(string Prompt) - affiche une boîte de dialogue invitant l'utilisateur à entrer une valeur ; renvoie la valeur de string saisie par l'utilisateur • Output(string Output) - écrit du texte à l'emplacement de sortie par défaut actuel ; durant: <ul style="list-style-type: none"> - Exécution normale du script, la sortie est écrite dans l'onglet "Script" de la fenêtre Sortie système - Débogage de Script , la sortie est écrite dans la fenêtre Débuguer - Utilisation de la console de script, la sortie est écrite sur la console • Prompt(string Prompt, long PromptType) - affiche une boîte de dialogue modale contenant le texte d' prompt et les types de boutons spécifiés ; renvoie la valeur ' valeur ' correspondant au bouton sur lequel l'utilisateur a cliqué
Valeurs PromptType	<ul style="list-style-type: none"> • promptOK = 1 • inviteOUINON = 2 • promptOUINONANNULER = 3 • promptOKCANCEL = 4
Valeurs PromptResult	<ul style="list-style-type: none"> • résultatOK = 1 • résultatAnnuler = 2 • résultatOui = 3 • résultatNon = 4
Session.Prompt Exemple	(VBScript) Si (Session.Prompt("Continue ?", promptYESNO) = resultYes) Alors...

Flux de travail

Les flux de travail valident le travail et les actions des utilisateurs par rapport à la politique et aux procédures de votre modèle, fournissant une approche robuste pour appliquer la politique de l'entreprise et renforcer les directives de développement de projet.

Les administrateurs de projet peuvent écrire des flux de travail pour gérer la façon dont les utilisateurs interagissent avec un modèle, comme la gestion de la sécurité, la conformité du personnel et l'accès au modèle, et la surveillance des modifications apportées par les utilisateurs. Les administrateurs peuvent également utiliser des workflows pour contrôler la capacité d'un utilisateur à modifier un élément de modèle, en tenant compte de facteurs tels que les droits d'accès, l'appartenance à un groupe et même la valeur d'une modification proposée.

Application des flux de travail

Considération	La description
Gouvernance du projet	<p>Une bonne gouvernance d'entreprise repose sur des directives de développement de projet bien rédigées et transparentes et sur la politique de l'entreprise.</p> <p>Un projet peut être compromis si les politiques et procédures appropriées sont mal comprises et ne sont pas suivies correctement - une gouvernance efficace peut être entravée par une erreur humaine et les coûts de récupération d'une conformité inadéquate des développeurs.</p>
Politiques, procédures et développement	<p>La politique et les procédures de l'entreprise peuvent être intégrées au processus de développement pour gérer les flux de travail, déterminer les droits d'accès, étendre les autorisations de sécurité basées sur les rôles et répondre aux événements de changement de propriété.</p> <p>Cette approche réduit les coûts de mise en conformité, améliore le développement collaboratif et vous donne l'assurance que les projets sont développés correctement du premier coup.</p> <p>Les équipes de développement peuvent adhérer aux directives des meilleures pratiques qui régissent la validation des modèles, la gestion des modifications, les contrôles d'accès et les principes généraux de développement.</p>

Options de flux de travail

Il existe deux options pour utiliser les scripts de workflow :

- VBScriptName
- JavaScript .

Le VBScript est une version plus ancienne et se limite à une série de commandes. La version JavaScript est une édition plus récente et permet un accès complet à toutes les fonctions d'automatisation.

Le JavaScript est documenté sous *EA_Connect* et les rubriques d'aide *Workflow Add-In Événements* .

Le VBScript est documenté dans la rubrique d'aide *Fonctions de script de workflow* .

Notes

- Les flux de travail sont disponibles dans les éditions Corporate , Unified et Ultimate d' Enterprise Architect

Fonctions de script de flux de travail

Note : À partir de la version 15.0 d' Enterprise Architect , les scripts de workflow VBScript sont disponibles mais obsolètes. Vous pouvez maintenant utiliser l'événement de modèle Enterprise Architect Add-In EA_Connect pour répondre aux événements Workflow Add-In , qui ont une gamme plus large de fonctionnalités et ne dépendent pas de Visual Basic.

Les scripts de workflow sont créés dans la fenêtre Scriptant , sous le type de groupe Workflow en tant que VBScripts. Ils sont exécutés par le moteur de workflow d' Enterprise Architect , pour gérer les entrées de l'utilisateur.

Vous pouvez utiliser une gamme de fonctions et de structures de données pour développer vos scripts.

Accéder

Ouvrez la fenêtre Scriptant en utilisant l'une des méthodes décrites ici, puis cliquez sur le bouton Nouveau groupe pour créer un nouveau groupe de scripts de workflow, avant de cliquer sur le bouton Nouveau script pour créer un nouveau script.

Ruban	Spécialiser > Outils > Scriptant
-------	----------------------------------

Fonctions de workflow et structures de données

Fonction	La description
Implémentation de scripts	<p>Lorsqu'un modèle est lancé, le moteur de workflow est initialisé avec les appartenances actuelles de l'utilisateur et du groupe ; ces informations déterminent qui peut accéder et modifier les parties d'un modèle donné.</p> <p>Lorsqu'un événement sélectionné se produit, le moteur de script est initialisé avec des valeurs comprenant le nom de l'auteur et les droits d'accès, ainsi que le nom de l'élément et les détails de la version.</p> <p>Le script de workflow implémente des règles régissant la gestion des modifications, le contrôle d'accès et la validation du modèle ; si un utilisateur tente d'apporter des modifications en violation de la politique de l'entreprise, le script refuse la mise à jour.</p> <p>L'utilisateur est informé de l'échec de la validation et l'activité est consignée.</p> <p>Ces rappels aident à renforcer la politique de l'entreprise, à réduire les erreurs humaines et à fournir à la direction des informations précieuses sur le projet.</p>
Fonctions pour la saisie utilisateur	<p>Ce sont des fonctions Enterprise Architect appelle pour valider et contrôler l'entrée de l'utilisateur.</p> <p>Pour chacune des fonctions appelées par Enterprise Architect , un ensemble d'objets est rempli.</p>
Fonctions pour créer une recherche	<p>Ce sont des fonctions Enterprise Architect appelle pour créer une recherche avec des tâches utilisateur.</p>
Workflow Data Structures Enterprise Architect	<p>Ce sont des objets de structure de données de flux de travail remplis par Enterprise Architect .</p>

remplit	
Structures de données de workflow que vous remplissez	Ce sont des objets de structure de données Workflow que vous pouvez remplir.
Fonctions que vous appelez	Ce sont des fonctions Enterprise Architect vous propose d'appeler.

Notes

- Si vous apportez des modifications à un script de workflow répertorié dans la fenêtre Scriptant , cliquez sur le bouton Actualiser les Scripts dans la barre d'outils de la fenêtre Scriptant pour recharger le script avec les modifications.
- Workflow Scriptant est disponible dans les éditions Corporate , Unified et Ultimate d' Enterprise Architect
- Workflow Scriptant nécessite que la sécurité de l'utilisateur soit activée pour fonctionner
- Vous avez besoin de l'autorisation "Admin Workflow" pour développer et gérer Scripts de workflow

Fonctions - Valider et contrôler l'entrée de l'utilisateur

Enterprise Architect appelle un certain nombre de fonctions pour valider et contrôler l'entrée de l'utilisateur. Pour chaque fonction un ensemble d'objets est rempli.

Valider/contrôler l'entrée de l'utilisateur

Fonction	Action
AllowPhaseUpdate(AncienneValeur, NewValue)	Valider une modification qu'un utilisateur a apportée à une phase. Valeur de retour : <ul style="list-style-type: none"> • True pour autoriser cet utilisateur à effectuer cette modification • False pour interdire le changement et revenir à la valeur précédente
AllowStatusUpdate(OldValue, NewValue)	Valider une modification qu'un utilisateur a apportée à un statut. Valeur de retour : <ul style="list-style-type: none"> • True pour autoriser cet utilisateur à effectuer cette modification • False pour interdire le changement et revenir à la valeur précédente
AllowTagUpdate(TagName, AncienneValeur, NewValue)	Valider une modification qu'un utilisateur a apportée à une Valeur Étiquetée . Valeur de retour : <ul style="list-style-type: none"> • True pour autoriser cet utilisateur à effectuer cette modification • False pour interdire le changement et revenir à la valeur précédente
AllowVersionUpdate(OldValue, NewValue)	Valider une modification qu'un utilisateur a apportée à une version. Valeur de retour : <ul style="list-style-type: none"> • True pour autoriser cet utilisateur à effectuer cette modification • False pour interdire le changement et revenir à la valeur précédente
CanEditPhase()	Activer ou désactiver le contrôle d'édition d'une phase Valeur de retour : <ul style="list-style-type: none"> • True pour permettre à cet utilisateur d'apporter des modifications en activant le contrôle • False pour désactiver complètement la modification de cette propriété en désactivant le contrôle
CanEditStatus()	Activez ou désactivez le contrôle pour modifier un statut. Valeur de retour : <ul style="list-style-type: none"> • True pour permettre à cet utilisateur d'apporter des modifications en activant le contrôle • False pour désactiver complètement la modification de cette propriété en désactivant le contrôle
CanEditTag(TagName)	Activer ou désactiver le contrôle d'édition d'une Valeur Étiquetée . Valeur de retour : <ul style="list-style-type: none"> • True pour permettre à cet utilisateur d'apporter des modifications en activant le contrôle

	<ul style="list-style-type: none"> • False pour désactiver complètement la modification de cette propriété en désactivant le contrôle
CanEditVersion()	<p>Activez ou désactivez le contrôle pour modifier une version.</p> <p>Valeur de retour :</p> <ul style="list-style-type: none"> • True pour permettre à cet utilisateur d'apporter des modifications en activant le contrôle • False pour désactiver complètement la modification de cette propriété en désactivant le contrôle
OnPreNewElement(TypeElement, élément stéréotype)	<p>Autoriser ou interdire la création de l'élément spécifié.</p> <p>Valeur de retour :</p> <ul style="list-style-type: none"> • True pour autoriser cet utilisateur à créer l'élément/le connecteur • False pour empêcher cet utilisateur de créer l'élément
SurPréNouveauConnecteur (TypeConnecteur, ConnectorSubType, connecteur stéréotype)	<p>Autoriser ou interdire la création du connecteur spécifié.</p> <p>Valeur de retour :</p> <ul style="list-style-type: none"> • Vrai pour permettre à cet utilisateur de créer l'élément/connecteur • False pour empêcher cet utilisateur de créer l'élément
PreAllowPhaseUpdate(AncienneValeur, NewValue)	<p>Déterminez quelles informations sont requises pour valider ce changement.</p> <p>Valeur de retour : liste séparée par des points-virgules des données supplémentaires requises pour valider ce changement.</p> <p>Type de données pris en charge :</p> <ul style="list-style-type: none"> • Tests - remplissez le tableau Tests dans l' object WorkflowContext
PreAllowStatusUpdate(Old Value, NewValue)	<p>Déterminez quelles informations sont requises pour valider ce changement.</p> <p>Valeur de retour : liste séparée par des points-virgules des données supplémentaires requises pour valider ce changement.</p> <p>Type de données pris en charge :</p> <p>Tests - remplissez le tableau Tests dans l' object WorkflowContext</p>
PreAllowTagUpdate(TagName, AncienneValeur, NewValue)	<p>Déterminez quelles informations sont requises pour valider ce changement.</p> <p>Valeur de retour : liste séparée par des points-virgules des données supplémentaires requises pour valider ce changement.</p> <p>Type de données pris en charge :</p> <p>Tests - remplissez le tableau Tests dans l' object WorkflowContext</p>
PreAllowVersionUpdate(AncienneValeur, NewValue)	<p>Déterminez quelles informations sont requises pour valider ce changement.</p> <p>Valeur de retour : liste séparée par des points-virgules des données supplémentaires requises pour valider ce changement.</p> <p>Type de données pris en charge :</p> <p>Tests - remplissez le tableau Tests dans l' object WorkflowContext</p>

Fonctions - Créer une recherche avec des tâches utilisateur

Ce sont des fonctions Enterprise Architect appelle pour créer une recherche avec des tâches utilisateur.

Les fonctions

Fonction	Action
GetWorkflowTasks	Décrivez les recherches que cet utilisateur doit exécuter . Valeur de retour : ignorée

Structures de données de flux de travail remplies

Ce sont les structures de données de flux de travail (objets) Enterprise Architect remplis.

Structures de données

Structure des données du flux de travail	La description
WorkflowUser	<p>Cet objet fournit des informations sur l'utilisateur actuellement connecté au modèle.</p> <p>Il est rempli par Enterprise Architect avant qu'une fonction ne soit appelée par Enterprise Architect ; il a ces propriétés :</p> <ul style="list-style-type: none"> • Nom d'utilisateur - le nom d'utilisateur pour la connexion au système (si vous utilisez l'authentification Windows , cela correspond au nom d'utilisateur Windows) • Prénom - tel qu'il se trouve dans la boîte de dialogue "Utilisateurs de sécurité" • Nom de famille - tel qu'il se trouve dans la boîte de dialogue "Utilisateurs de sécurité" • Nom complet - la combinaison <Firstname> <Surname> (la forme Enterprise Architect utilise pour les champs 'Auteur' et similaires) • Département - le département dans lequel l'utilisateur travaille, tel qu'il se trouve dans la boîte de dialogue "Utilisateurs de sécurité" <p>Appels : cet objet appelle la IsMemberOf(GroupName) .</p>
WorkflowContext	<p>Cet objet fournit des informations sur l' object actuellement en contexte.</p> <p>Il est rempli par Enterprise Architect avant que toute recherche à l'exception de exécuter ne soit effectuée ; il a ces propriétés :</p> <ul style="list-style-type: none"> • MetaType - le type de l' object actuel, soit un type principal d' Enterprise Architect , soit un métatype spécifié par le profil • Nom - tel qu'il se trouve dans la boîte de dialogue " Propriétés " de l' object • Statut - tel qu'il se trouve dans la boîte de dialogue de l' object ' Propriétés ' • Phase - telle qu'elle se trouve dans la boîte de dialogue de l' object ' Propriétés ' • Version - telle qu'elle se trouve dans la boîte de dialogue de l' object ' Propriétés ' • Stéréotypes - un tableau de chaînes pour les stéréotypes appliqués à cet object • Tags - un tableau de Valeur Étiquetées , fournissant : <ul style="list-style-type: none"> - Nom - le nom Valeur Étiquetée - Valeur - la Valeur Étiquetée valeur • Tests - un éventail de tests ; rempli uniquement lors d'un appel Allow* après que l'appel PreAllow* a spécifié que des tests sont requis ; fournit ces détails, tels qu'ils se trouvent dans la fenêtre Test Cases : <ul style="list-style-type: none"> - Nom - Statut - Exécuté par - Vérifié par - TestClass - Type d'essai <p>Appels : cet objet appelle la fonction TagValue(TagName).</p>

Les fonctions

Fonction	Action
IsMemberOf(GroupName)	Vérifiez l'appartenance au groupe de l'utilisateur actuel. Valeur de retour : retourne la valeur True si l'utilisateur courant est membre du groupe avec le nom spécifié.
TagValue(TagName)	Obtenir la valeur d'une balise nommée. Valeur de retour : renvoie la valeur de la première Valeur Étiquetée de ce nom, ou une string vide s'il n'existe aucune Valeur Étiquetée de ce nom.

Structures de données de workflow que vous remplissez

Il s'agit des structures de données de workflow (objets) que vous pouvez remplir.

Structures de données

Structure des données du flux de travail	La description
WorkflowStatus	<p>Utilisez cette structure de données pour fournir des informations sur l'état de l'objet .</p> <ul style="list-style-type: none"> • LogEntry - défini sur True ou False pour indiquer si un élément de log doit être enregistré ou non • Raison - indiquez quelle raison doit être enregistrée dans le log • Action - indiquez comment afficher le message du log ; les valeurs valides sont : MessageBox, StatusBar et Output (par défaut)
WorkflowSearches	<p>Utilisez cette structure de données pour fournir un tableau de recherches. Utilisez Redim WorkflowSearches (x) pour spécifier le nombre de recherches fournies.</p> <p>Chaque recherche a ces attributs :</p> <ul style="list-style-type: none"> • Nom - le nom de cette recherche • Groupe - le nom du groupe sous lequel cette recherche doit apparaître dans la liste déroulante "Rechercher" • ID - le GUID pour cette recherche • Tâches - le tableau des tâches recherchées par cette recherche ; une entrée décrit comment trouver tous les objets requis pour répondre à une tâche particulière : <ul style="list-style-type: none"> - Nom - le nom de la tâche, tel qu'affiché dans la recherche de Modèle voir; les recherches de workflow sont regroupées par ce champ par défaut - Conditions - un tableau de conditions, qui doivent toutes être mises en correspondance pour un objet à inclure dans cette tâche ; une condition est une comparaison de un seul champ à une valeur : <ul style="list-style-type: none"> - Colonne - le nom du champ - Opérateur - types d'opérateurs, soit = (fournir uniquement des valeurs correspondantes) ou <> (fournir uniquement des valeurs qui ne correspondent pas) - Valeur - si elle contient une virgule, la string est traitée comme un liste de valeurs séparées par des virgules à comparer ; sinon la string est une valeur unique à comparer

Fonctions que vous appelez

indéfini

Les fonctions


Fonction	Action
NewSearch (nom, groupe, guid, nombre de tâches)	indéfini
NewTask(nom, conditioncount)	indéfini
NouvelleCondition(colonne, opérateur, valeur)	indéfini
SetLastError(message, outputMethod)	indéfini

Débogage de Script


Le débogage de script aide au développement et à la maintenance des scripts modèles et à la surveillance de leur activité au moment de l'exécution. Lors du débogage d'un script, vous pouvez :

- Contrôlez le flux d'exécution à l'aide des boutons ' Déboguer ', ' Step Over ', ' Step Into ', ' Sortir ' et ' Stop Script ' de la barre d'outils de l' Éditeur de Script
- Set Points d'Arrêt , Marqueurs d'Enregistrement et Marqueurs Point de Trace
- Utilisez la fenêtre Déboguer pour afficher la sortie générée par le script
- Utilisez la fenêtre Variables locales pour inspecter les valeurs des variables, y compris les objets de l'interface d'automatisation
- Utilisez la fenêtre Enregistrer et analyser pour enregistrer un Séquence de diagramme de l'exécution du script

Accéder

Ruban	Spécialiser > Outils > Script Bibliothèque > cliquez-droit sur [nom du script] > Script Débogage
Autre	Barre d'outils de la fenêtre Éditeur de Script : Cliquez sur l'icône de la barre d'outils 

Commencer à déboguer un script de modèle

Marcher	Action
1	Ouvrez un modèle de script dans l' Éditeur de Script .
2	Définissez tous les Points d'Arrêt sur la ou les lignes de code appropriées.
3	Cliquez sur l'icône de la barre d'outils  (Déboguer).

Notes

- Le débogage de script est pris en charge pour VBScript, JScript et JavaScript
- VBScript et JScript nécessitent que Microsoft Process Débogueur Manager soit installé sur la machine locale ; ceci est disponible via divers produits Microsoft, y compris le "Microsoft Script Débogueur " gratuit
- Points d'Arrêt ne sont pas enregistrés pour les scripts et ne persisteront pas lors de la prochaine ouverture du script
- Lors du débogage, la sortie du script est redirigée vers la fenêtre Déboguer

