

# TAMUCALC 6.2 Command Reference

COPYRIGHT ©2009 by Art Belmonte  
Department of Mathematics, Texas A&M University

Monday, 06 July 2009

1. See <http://calclab.math.tamu.edu/~belmonte/TAMUCALC/TAMUCALC.html> for instructions on downloading and installing the *TAMUCALC* package, refreshing libraries, and using preactivated templates.
  2. When starting a new problem, run the **activate** command under *atm\_tamucalc62* in the libraries portion of the catalog.
  3. It is not necessary to reactivate the package on subsequent *pages* of a given problem—only if you start a *new* problem.
  4. Rather than manual [re]activation, you will find preactivated templates much easier and faster to use!
- 

## ABOUT

<b>Description</b>	The <b>about</b> program displays TAMUCALC credits.
<b>Access</b>	<b>h.about</b> library shortcut in the <i>HELP</i> group
<b>Syntax</b>	<b>h.about()</b>
<b>Input</b>	(none)
<b>Output</b>	display of TAMUCALC credits
<b>Example</b>	The command returns the following display. TAMUCALC 6.2 Monday, 06 July 2009 Calculus package for TI-Nspire CAS © 2009, Art Belmonte; under GPLv3, 2007 EMAIL: Art.Belmonte@math.tamu.edu
<b>See Also</b>	Information regarding the package is comprised of this command reference document and online streaming videos.

---

## ANGVEC

<b>Description</b>	The <b>angvec</b> function computes the exact angle between two vectors using the current angle mode.
<b>Access</b>	<b>v.angvec</b> library shortcut in the <i>vectors</i> group
<b>Syntax</b>	<b>v.angvec(v,w)</b>
<b>Input</b>	<b>v</b> : a vector having two or more components <b>w</b> : a vector with the same number of components as <b>v</b>
<b>Output</b>	the exact angle $\theta = \cos^{-1}\left(\frac{\mathbf{v} \cdot \mathbf{w}}{\ \mathbf{v}\  \ \mathbf{w}\ }\right)$ between <b>v</b> and <b>w</b> in the current angle mode
<b>Examples</b>	<b>v.angvec</b> ([3, 1, -1], [1, -1, 2]) returns $\frac{\pi}{2}$ in <b>RADIAN</b> mode. <b>v.angvec</b> ([3, 1, -1], [1, -1, 2])▶ <b>DD</b> returns 90°.
<b>See Also</b>	▶ <b>DD</b> in TI-Nspire CAS documentation

---

---

## CA

**Description** The **ca** function computes the acceleration of a position curve.

**Access** **p.ca** library shortcut in the *primes* group

**Syntax** **p.ca(g,t)**

**Input** **g**: position vector expression  
**t**: independent variable (time)

**Output** the expression for acceleration,  $\mathbf{a} = \mathbf{g}''$

**Example** **p.ca**( $[t, 2t^2, 3t^3], t$ ) returns  $[0, 4, 18t]$ .

**See Also** **CS, CV**

---

## CAN

**Description** The **can** function computes the normal component of acceleration of a position curve.

**Access** **p.can** library shortcut in the *primes* group

**Syntax** **p.can(g,t)**

**Input** **g**: position vector expression  
**t**: independent variable (time)

**Output** the expression for the normal component of acceleration,  $a_N = \|\mathbf{T}'\| \|\mathbf{g}'\|$ , where  $\mathbf{T} = \frac{\mathbf{g}'}{\|\mathbf{g}'\|}$

**Example** **p.can**( $[t, 2t, t^2], t$ ) returns  $\frac{2\sqrt{5}}{\sqrt{4t^2 + 5}}$ .

**See Also** **CA, CAT**

---

## CAT

**Description** The **cat** function computes the tangential component of acceleration of a position curve.

**Access** **p.cat** library shortcut in the *primes* group

**Syntax** **p.cat(g,t)**

**Input** **g**: position vector expression  
**t**: independent variable (time)

**Output** the expression for the tangential component of acceleration,  $a_T = \frac{\mathbf{g}' \cdot \mathbf{g}''}{\|\mathbf{g}'\|}$

**Example** **p.cat**( $[t, 2t, t^2], t$ ) returns  $\frac{4t}{\sqrt{4t^2 + 5}}$ .

**See Also** **CA, CAN**

---

---

## CB

**Description** The **cb** function computes the unit binormal vector to a position curve.

**Access** **c.cb** library shortcut in the *curves* group

**Syntax** **c.cb(g,t)**

**Input** **g**: position vector expression  
**t**: independent variable (time)

**Output** the unit binormal vector  $\mathbf{B} = \frac{\mathbf{g}' \times \mathbf{g}''}{\|\mathbf{g}' \times \mathbf{g}''\|}$  to the curve

**Example** **c.cb**([cos(t), sin(t), t], t) returns  $\left[ \frac{\sqrt{2} \sin t}{2}, -\frac{\sqrt{2} \cos t}{2}, \frac{\sqrt{2}}{2} \right]$ .

**See Also** **CN, CT**

---

## CJ

**Description** The **cj** function computes the jerk of a position curve.

**Access** **p.cj** library shortcut in the *primes* group

**Syntax** **p.cj(g,t)**

**Input** **g**: position vector expression  
**t**: independent variable (time)

**Output** the expression for jerk,  $\mathbf{g}'''$

**Example** **p.cj**([t, 2t<sup>2</sup>, 3t<sup>3</sup>], t) returns [0, 0, 18].

**See Also** **CA**

---

## CK

**Description** The **ck** function computes the curvature of a position curve.

**Access** **c.ck** library shortcut in the *curves* group

**Syntax** **c.ck(g,t)**

**Input** **g**: position vector expression  
**t**: independent variable (time)

**Output** the curvature  $\kappa = \frac{\|\mathbf{T}'\|}{\|\mathbf{g}'\|}$  of the curve, where  $\mathbf{T} = \frac{\mathbf{g}'}{\|\mathbf{g}'\|}$

**Example** **c.ck**([t, 2t, t<sup>2</sup>], t) returns  $\frac{2\sqrt{5}}{(4t^2 + 5)^{3/2}}$ .

**See Also** **CTAU**

---

---

## CL

**Description** The **cl** function computes the arc length of a position curve.

**Access** **c.cl** library shortcut in the *curves* group

**Syntax** **c.cl(*g,t,a,b*)**

**Input** **g**: position vector expression  
**t**: independent variable (time)  
**a**: lower bound of *t*-range  
**b**: upper bound of *t*-range

**Output** The arc length  $\int_a^b \|\mathbf{g}'\| dt$  of the curve

**Example** **c.cl**( $[t, 2t, t^2], t, 0, 1$ ) returns  $\frac{5}{8} \ln 5 + \frac{3}{2} \approx 2.51$ .

**See Also** **CK, CTAU**

---

## CLINE2PT

**Description** The **cline2pt** function finds Cartesian equation(s) of the line between two points in the *xy*-plane.

**Access** **g.cline2pt** library shortcut in the *geometry* group

**Syntax** **g.cline2pt(*a,b*)**

**Input** **a**: a point on the line (as a position vector)  
**b**: a different point on the line (as a position vector)

**Output** Cartesian equation(s) of the line

**Examples** **g.cline2pt**( $[5, 2], [8, 2]$ ) returns  $y = 2$ .  
**g.cline2pt**( $[3, 6], [3, -4]$ ) returns  $x = 3$ .  
**g.cline2pt**( $[2, 3], [1, 1]$ ) returns  $\begin{bmatrix} x = \frac{1}{2}y + \frac{1}{2} \\ y = 2x - 1 \end{bmatrix}$ .

**See Also** **LINPTDIR, PLINE2PT**

---

## CN

**Description** The **cn** function computes the unit normal vector to a position curve.

**Access** **c.cn** library shortcut in the *curves* group

**Syntax** **c.cn(*g,t*)**

**Input** **g**: position vector expression  
**t**: independent variable (time)

**Output** the unit normal vector  $\mathbf{N} = \frac{\mathbf{T}'}{\|\mathbf{T}'\|}$  to the curve, where  $\mathbf{T} = \frac{\mathbf{g}'}{\|\mathbf{g}'\|}$

**Example** **c.cn**( $[\cos(t), \sin(t), t], t$ ) returns  $[-\cos t, -\sin t, 0]$ .

**See Also** **CB, CT**

---

---

## COMMANDS1

**Description** The **commands1** function displays a table of primary TAMUCALC commands.

**Access** **h.commands1** library shortcut in the *HELP* group

**Syntax** **h.commands1()**

**Input** (none)

**Output** a listing of primary TAMUCALC commands

**Examples** Invoking **h.commands1()** displays primary TAMUCALC commands by group.

<b>c.</b>	<i>cb, ck, cl, cn, ct, ctau</i>
<b>d.</b>	<i>dpthline, dptplane, ftli</i>
<b>d.</b>	<i>jmatrix, lapl, pot</i>
<b>f.</b>	<i>curl, div, grad, hess, invfunc, lpmd</i>
<b>g.</b>	<i>cline2pt, linptdir, plane3pt</i>
<b>g.</b>	<i>planeint, planenpt, pline2pt</i>
<b>i.</b>	<i>compz, lis, liv, orient, sis, siv</i>

**See Also** **COMMANDS2** (for secondary TAMUCALC commands)

---

## COMMANDS2

**Description** The **commands2** function displays a table of secondary TAMUCALC commands.

**Access** **h.commands2** library shortcut in the *HELP* group

**Syntax** **h.commands2()**

**Input** (none)

**Output** a listing of secondary TAMUCALC commands

**Examples** Invoking **h.commands2()** displays secondary TAMUCALC commands by group.

<b>a.</b>	<i>diff, equate, int1, intg, sumv, u2u</i>
<b>h.</b>	<i>about, commands1, commands2</i>
<b>h.</b>	<i>entryaids, groups1, groups2</i>
<b>m.</b>	<i>int1, int2, int3, sum1, sum2, sum3</i>
<b>p.</b>	<i>ca, can, cat, cj, cs, cv</i>
<b>v.</b>	<i>angvec, comp, orth, perp, proj, stp</i>

**See Also** **COMMANDS1** (for primary TAMUCALC commands)

---

---

## COMP

**Description** The **comp** function computes the scalar projection of one vector onto another.

**Access** **v.comp** library shortcut in the *vectors* group

**Syntax** **v.comp(v,w)**

**Input** **v**: a vector having two or more components  
**w**: a vector with the same number of components as **v**

**Output** The scalar projection  $\text{comp}_v \mathbf{w} = \frac{\mathbf{v} \cdot \mathbf{w}}{\|\mathbf{v}\|}$  of the vector **w** onto the vector **v**

**Example** **v.comp**([6, 1, -2], [0, 4, 7]) returns  $-\frac{10\sqrt{41}}{41} \approx -1.56$ .

**See Also** **ORTH, PROJ**

---

## COMPZ

**Description** The **compz** function yields the composition of two functions.

**Access** **i.compz** library shortcut in the *integrals* group

**Syntax** **i.compz(f,v,g)**

**Input** **f**: an expression that signifies the outer function  
**v**: a row vector (usually of two or more elements): the variables in which **f** is expressed  
**g**: a row vector (with the same number of elements as **v**) that signifies the inner function

**Output** The composition **f**∘**g** evaluated at **v**; i.e.,  $f(g(\mathbf{v}))$

**Examples**  $x^3y^2z \rightarrow \mathbf{f}: [2t+3, t^2-1, t^3] \rightarrow \mathbf{g}: \mathbf{i.compz}(\mathbf{f}, [x, y, z], \mathbf{g})$  returns  $(2t+3)^3(t^2-1)^2t^3$ .  
 $[\sin(x \cdot y \cdot z), x^2y, z^2e^{x/5}] \rightarrow \mathbf{w}: [u, \cos(v), 2\sin(v)] \rightarrow \mathbf{s}: \mathbf{i.compos}(\mathbf{w}, [x, y, z], \mathbf{s})$  returns  $[\sin(2u \sin v \cos v), u^2 \cos v, 4e^{u/5} \sin^2 v]$ .  
 $\sqrt{x} \rightarrow \mathbf{f}: [2t+1] \rightarrow \mathbf{g}: \mathbf{i.compz}(\mathbf{f}, [x], \mathbf{g})$  returns  $\sqrt{2t+1}$ .

**See Also** | (the substitution facility) in TI-Nspire CAS documentation

---

## CS

**Description** The **cs** function computes the speed of a position curve.

**Access** **p.cs** library shortcut in the *primes* group

**Syntax** **p.cs(g,t)**

**Input** **g**: position vector expression  
**t**: independent variable (time)

**Output** the expression for speed,  $\|\mathbf{g}'\|$ , the magnitude of velocity

**Example** **p.cs**([t, 2t<sup>2</sup>, 3t<sup>3</sup>], t) returns  $\sqrt{81t^4 + 16t^2 + 1}$ .

**See Also** **CA, CV**

---

---

## CT

**Description** The **ct** function computes the unit tangent vector to a position curve.

**Access** **c.ct** library shortcut in the *curves* group

**Syntax** **c.ct(g,t)**

**Input** **g**: position vector expression  
**t**: independent variable (time)

**Output** the unit tangent vector  $\mathbf{T} = \frac{\mathbf{g}'}{\|\mathbf{g}'\|}$  to the curve

**Example** **c.ct**([cos(t), sin(t), t], t) returns  $\left[-\frac{\sqrt{2}\sin t}{2}, \frac{\sqrt{2}\cos t}{2}, \frac{\sqrt{2}}{2}\right]$ .

**See Also** **CB, CN**

---

## CTAU

**Description** The **ctau** function computes the torsion of a position curve.

**Access** **c.ctau** library shortcut in the *curves* group

**Syntax** **c.ctau(g,t)**

**Input** **g**: position vector expression  
**t**: independent variable (time)

**Output** the torsion  $\tau = \frac{(\mathbf{g}' \times \mathbf{g}'') \cdot \mathbf{g}'''}{\|\mathbf{g}' \times \mathbf{g}''\|^2}$  of the curve

**Example** **c.ctau**([sin(4t), 6t<sup>2</sup>, cos(4t)], t) returns  $\frac{48t}{144t^2 + 25}$ .

**See Also** **CK**

---

## CURL

**Description** The **curl** function computes the curl (rotational) of a vector field.

**Access** **f.curl** library shortcut in the *functions* group

**Syntax** **f.curl(w)**

**Input** **w**: A vector field having 3 components

**Output** the curl  $\vec{\nabla} \times \mathbf{w} = \det \begin{bmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ \partial/\partial x & \partial/\partial y & \partial/\partial z \\ p & q & r \end{bmatrix}$  of the vector field  $\mathbf{w} = [p, q, r]$

**Example** **f.curl**([x<sup>2</sup>y, yz<sup>2</sup>, x<sup>2</sup>z]) returns [-2yz, -2xz, -x<sup>2</sup>].

**See Also** **DIV, LAPL**

---

---

## CV

**Description** The **cv** function computes the velocity of a position curve.

**Access** **p.cv** library shortcut in the *primes* group

**Syntax** **p.cv(g,t)**

**Input** **g**: position vector expression  
**t**: independent variable (time)

**Output** the expression for velocity,  $\mathbf{v} = \mathbf{g}'$

**Example** **p.cv**( $[t, 2t^2, 3t^3], t$ ) returns  $[1, 4t, 9t^2]$ .

**See Also** **CA, CS**

---

## DIFF

**Description** The **diff** function computes a first or higher order derivative inline.

**Access** **a.diff** library shortcut in the *auxiliary* group

**Syntax** **a.diff(f,v)**

**Input** **f**: a scalar, vector, or matrix expression expression in **v**  
**v**: variable(s) with respect to which to (sequentially) differentiate

**Output** the derivative of **f** with respect to the given variable(s)

**Examples** **a.diff**( $x^5, x$ ) returns  $\frac{d}{dx}(x^5) = 5x^4$ .  
**a.diff**( $[x^4, x^6, x^8], x$ ) returns  $\frac{d}{dx}([x^4, x^6, x^8]) = [4x^3, 6x^5, 8x^7]$ .  
**a.diff**( $e^{5x} \ln(y^2 + 9), [x, y]$ ) returns  $\frac{\partial^2}{\partial y \partial x}(e^{5x} \ln(y^2 + 9)) = \frac{10e^{5x}y}{y^2 + 9}$ .  
**a.diff**( $[\sin(3x), x^7, e^{4x}], [x, x]$ ) returns  $\frac{d^2}{dx^2}([\sin(3x), x^7, e^{4x}]) = [-9 \sin(3x), 42x^5, 16e^{4x}]$ .

**See Also** **INTG** [to compute an inline indefinite integral (antiderivative),  $\int f dx$ ]

---

## DIV

**Description** The **div** function computes the divergence of a vector field.

**Access** **f.div** library shortcut in the *functions* group

**Syntax** **f.div(w)**

**Input** **w**: a vector field with three components

**Output** the divergence  $\vec{\nabla} \cdot \mathbf{w} = \frac{\partial p}{\partial x} + \frac{\partial q}{\partial y} + \frac{\partial r}{\partial z}$  of the vector field  $\mathbf{w} = [p, q, r]$

**Example** **f.div**( $[x^2y, yz^2, x^2z]$ ) returns  $x^2 + 2xy + z^2$ .

**See Also** **CURL, LAPL**

---

---

## DPTHLINE

<b>Description</b>	The <b>dpthline</b> function computes the distance from a point to a (hyper)line.
<b>Access</b>	<b>d.dpthline</b> library shortcut in the <i>distances</i> group
<b>Syntax</b>	<b>d.dpthline(<i>p</i>,<i>a</i>,<i>b</i>)</b>
<b>Input</b>	<b><i>p</i></b> : A point [not on the (hyper)line] as a position vector <b><i>a</i></b> : a point on the (hyper)line as a position vector <b><i>b</i></b> : a different point on the (hyper)line as a position vector
<b>Output</b>	the distance $d$ from the point <b><i>p</i></b> to the (hyper)line containing <b><i>a</i></b> and <b><i>b</i></b> , given by $d = \ \text{orth}_{\mathbf{v}}\mathbf{w}\ $ , where $\mathbf{v} = \vec{ab}$ and $\mathbf{w} = \vec{ap}$
<b>Example</b>	<b>d.dpthline</b> ([1, 0, -1], [5, 0, 1], [4, 3, 3]) returns $2\sqrt{5} \approx 4.47$ .
<b>See Also</b>	<b>DPTPLANE, ORTH</b>

---

## DPTPLANE

<b>Description</b>	The <b>dptplane</b> function computes the distance from a point to a plane.
<b>Access</b>	<b>d.dptplane</b> library shortcut in the <i>distances</i> group
<b>Syntax</b>	<b>d.dptplane(<i>p</i>,<i>a</i>,<i>b</i>,<i>c</i>)</b>
<b>Input</b>	<b><i>p</i></b> : a point (not on the plane) as a position vector <b><i>a</i></b> : a point on the plane as a position vector <b><i>b</i></b> : a different point on the plane as a position vector <b><i>c</i></b> : a third point on the plane as a position vector; <b><i>a</i></b> , <b><i>b</i></b> , and <b><i>c</i></b> are noncollinear
<b>Output</b>	The distance $d$ from the point <b><i>p</i></b> to the plane containing <b><i>a</i></b> , <b><i>b</i></b> , and <b><i>c</i></b> , given by $d = \ \text{proj}_{\mathbf{v}}\mathbf{w}\  =  \text{comp}_{\mathbf{v}}\mathbf{w} $ , where $\mathbf{v} = \vec{ab} \times \vec{ac}$ and $\mathbf{w} = \vec{ap}$
<b>Example</b>	<b>d.dptplane</b> ([3, -2, 7], [1, 0, 1], [2, 1, 3], [1, 2, 13]) returns $\frac{26\sqrt{53}}{53} \approx 3.57$ .
<b>See Also</b>	<b>COMP, DPTHLINE, PROJ</b>

---

---

## ENTRYAIDS

<b>Description</b>	The <b>entryaids</b> program explains how to set up library shortcuts, inline commands, or both.
<b>Access</b>	<b>h.entryaids</b> library shortcut in the <i>HELP</i> group
<b>Syntax</b>	<b>h.entryaids()</b>
<b>Input</b>	(none)
<b>Output</b>	Help on the <b>activate</b> and <b>setup</b> commands in the <i>atm_tamucalc62</i> library.
<b>Example</b>	Invoking <b>h.entryaids()</b> displays information to this effect. “In the library <i>atm_tamucalc62</i> ... <b>activate()</b> checks various system settings and runs <b>setup(1)</b> . <b>setup(1)</b> creates library shortcuts (only). <b>setup(2)</b> creates inline commands (only). <b>setup(3)</b> creates both types of entry aids.”
<b>See Also</b>	Be sure to view online streaming videos that illustrate the use of library shortcuts and inline commands. Basically, shortcuts provide quicker entry when using the TI-Nspire CAS Handheld, whereas inline commands are faster for touch typists using the TI-Nspire CAS Computer Software or TI-Nspire CAS Teacher’s Edition. (Only instructors demonstrating TAMUCALC to students might have occasion to use <i>both</i> typing aids!)

---

## EQUATE

<b>Description</b>	The <b>equate</b> function constructs simultaneous equations from the respective components of two vectors.
<b>Access</b>	<b>a.equate</b> library shortcut in the <i>auxiliary</i> group
<b>Syntax</b>	<b>a.equate(v,w)</b>
<b>Input</b>	<b>v</b> : a vector <b>w</b> : another vector with the same number of components as <b>v</b> .
<b>Output</b>	simultaneous equations joined with <b>ands</b> ; possibly simplified
<b>Example</b>	<b>a.equate</b> ( $[x,y,z]$ , $[3 \cos(t), 2 \sin(t), 5t]$ ) returns $x = 3 \cos(t)$ and $y = 2 \sin(t)$ and $z = 5t$ .
<b>See Also</b>	(the substitution facility) in TI-Nspire CAS documentation

---

---

## FTLI

<b>Description</b>	The <b>ftli</b> function computes the line integral of a conservative vector field $\mathbf{w}$ by applying the Fundamental Theorem for Line Integrals.
<b>Access</b>	<b>d.ftli</b> library shortcut in the <i>derivatives</i> group
<b>Syntax</b>	<b>d.ftli(<math>f, \mathbf{v}, \mathbf{g}, t, \mathbf{a}, \mathbf{b}</math>)</b>
<b>Input</b>	$f$ : a scalar expression that represents a potential function for the conservative vector field $\mathbf{w}$ $\mathbf{v}$ : a row vector of variables: the independent variables in which $f$ is expressed $\mathbf{g}$ : a vector expression that represents a parameterization for the curve $C$ $t$ : the variable of integration (typically $t$ ) $\mathbf{a}$ : lower bound of the variable of integration in the line integral $\mathbf{b}$ : upper bound of the variable of integration in the line integral
<b>Output</b>	The value of the line integral, $\int_C \mathbf{w} \cdot d\mathbf{g} = \int_C \vec{\nabla} f \cdot d\mathbf{g} = f(\mathbf{g}(b)) - f(\mathbf{g}(a))$ .
<b>Example</b>	<b>d.pot</b> ( $[4xe^z, \cos(y), 2x^2e^z], [x, y, z]$ ) $\rightarrow f$ yields $2x^2e^z + \sin y$ . Then $[t, t^2, t^4] \rightarrow \mathbf{g}$ : <b>d.ftli</b> ( $f, [x, y, z], \mathbf{g}, t, 0, 1$ ) returns $2e + \sin 1 \approx 6.28$ .
<b>See Also</b>	<b>POT</b>

---

## GRAD

<b>Description</b>	The <b>grad</b> function computes the gradient vector of a scalar function.
<b>Access</b>	<b>f.grad</b> library shortcut in the <i>functions</i> group
<b>Syntax</b>	<b>f.grad(<math>f, \mathbf{v}</math>)</b>
<b>Input</b>	$f$ : A scalar expression $\mathbf{v}$ : a row vector of independent variables
<b>Output</b>	The gradient vector $\vec{\nabla} f = \left[ \frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \dots, \frac{\partial f}{\partial x_n} \right]$ ; typically $\left[ \frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right]$ or $\left[ \frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z} \right]$ .
<b>Example</b>	<b>f.grad</b> ( $x^2 \sin(y) + 4xz^3, [x, y, z]$ ) returns $\{2x \sin y + 4z^3, x^2 \cos y, 12xz^2\}$ .
<b>See Also</b>	<b>HESS</b>
<b>Note</b>	A <i>list</i> (as opposed to a vector) is returned to facilitate work involved in computing extrema (maxima and minima).

---

---

## GROUPS1

**Description** The **groups1** function displays short descriptions of primary TAMUCALC groups.

**Access** **h.groups1** library shortcut in the *HELP* group

**Syntax** **h.groups1()**

**Input** (none)

**Output** short descriptions of primary TAMUCALC groups

**Examples** Invoking **h.groups1()** displays the following short descriptions of primary TAMUCALC groups.

<b>c.</b>	curves (differential geometry)
<b>d.</b>	derivative objects & theorems, distances
<b>f.</b>	functions + vector & matrix derivatives
<b>g.</b>	geometry of lines and planes
<b>i.</b>	line and surface integrals

**See Also** **GROUPS2** (for descriptions of secondary TAMUCALC groups)

---

## GROUPS2

**Description** The **groups2** function displays short descriptions of secondary TAMUCALC groups.

**Access** **h.groups2** library shortcut in the *HELP* group

**Syntax** **h.groups2()**

**Input** (none)

**Output** short descriptions of secondary TAMUCALC groups

**Examples** Invoking **h.groups2()** displays the following short descriptions of secondary TAMUCALC groups.

<b>a.</b>	auxiliary calculus/utility commands
<b>h.</b>	HELP with command groups and entry aids
<b>m.</b>	multiple integrals & approximations
<b>p.</b>	primes (derivatives of position)
<b>v.</b>	vector commands

**See Also** **GROUPS1** (for descriptions of primary TAMUCALC groups)

---

---

## HESS

**Description** The **hess** function computes the Hessian matrix of a scalar function.

**Access** **f.hess** library shortcut in the *functions* group

**Syntax** **f.hess(f,v)**

**Input** **f**: a scalar expression  
**v**: A row vector of independent variables

**Output** the Hessian matrix  $H = \begin{bmatrix} f_{x_1x_1} & \cdots & f_{x_1x_n} \\ \vdots & \ddots & \vdots \\ f_{x_nx_1} & \cdots & f_{x_nx_n} \end{bmatrix}$ ; typically  $H = \begin{bmatrix} f_{xx} & f_{xy} \\ f_{yx} & f_{yy} \end{bmatrix}$ .

**Example** **f.hess**( $x^3 + 4x \cdot y - y^2$ ,  $[x,y]$ ) returns  $\begin{bmatrix} 6x & 4 \\ 4 & -2 \end{bmatrix}$ .

**See Also** **GRAD, LPMD**

---

## INT1

**Description** The **int1** function computes the single integral  $\int_a^b f(x) dx$  in an inline fashion.

**Access** **m.int1** library shortcut in the *multiple* group

**Syntax** **m.int1(f,x,a,b)**

**Input** **f**: a scalar, vector, or matrix expression expression in **x**  
**x**: variable of integration  
**a**: lower limit of integration  
**b**: upper limit of integration

**Output** the definite integral  $\int_a^b f(x) dx$

**Examples** **m.int1**( $x^2 + 1$ ,  $x$ , 0, 2) returns  $\int_0^2 x^2 + 1 dx = \frac{14}{3} \approx 4.67$ .

**m.int1**( $[x,x^2,x^3]$ ,  $x$ ,  $a$ ,  $b$ ) returns  $\int_a^b [x,x^2,x^3] dx = \left[ \frac{b^2 - a^2}{2}, \frac{b^3 - a^3}{3}, \frac{b^4 - a^4}{4} \right]$ .

**See Also** **INT2, INT3** (for inline double and triple integrals, respectively)

**Note** The **a.int1** library shortcut in the *auxiliary* group also provides access to this command.

---

---

## INT2

<b>Description</b>	The <b>int2</b> function computes the double integral $\int_c^d \int_a^b f(u, v) \, dudv$ in an inline fashion.
<b>Access</b>	<b>m.int2</b> library shortcut in the <i>multiple</i> group
<b>Syntax</b>	<b>m.int2(f, u, a, b, v, c, d)</b>
<b>Input</b>	<b>f</b> : a scalar, vector, or matrix expression expression in <b>u</b> and <b>v</b> <b>u</b> : inner variable of integration <b>a</b> : lower limit of the inner variable <b>b</b> : upper limit of the inner variable <b>v</b> : outer variable of integration <b>c</b> : lower limit of the outer variable <b>d</b> : upper limit of the outer variable
<b>Output</b>	the definite integral $\int_c^d \int_a^b f(u, v) \, dudv$
<b>Examples</b>	<b>m.int2</b> ( $x^2 + y^2, y, 0, 2 - 2x, x, 0, 1$ ) returns $\int_0^1 \int_0^{2-2x} x^2 + y^2 \, dydx = \frac{5}{6} \approx 0.83$ . $\frac{1}{5/6} \cdot \mathbf{m.int2}((x^2 + y^2) \cdot [x, y], y, 0, 2 - 2x, x, 0, 1)$ returns $\left[ \frac{7}{25}, \frac{26}{25} \right] = [0.28, 1.04]$ .
<b>See Also</b>	<b>INT1</b> , <b>INT3</b> (for inline single and triple integrals, respectively)
<b>Note</b>	As the examples show, it is unnecessary for the variables of integration to be in alphabetical order.

---

## INT3

<b>Description</b>	The <b>int3</b> function computes the triple integral $\int_r^s \int_c^d \int_a^b f(u, v, w) \, dudvdw$ in an inline fashion.
<b>Access</b>	<b>m.int3</b> library shortcut in the <i>multiple</i> group
<b>Syntax</b>	<b>m.int3(f, u, a, b, v, c, d, w, r, s)</b>
<b>Input</b>	<b>f</b> : a scalar, vector, or matrix expression expression in <b>u</b> , <b>v</b> , and <b>w</b> <b>u</b> : inner variable of integration <b>a</b> : lower limit of the inner variable <b>b</b> : upper limit of the inner variable <b>v</b> : middle variable of integration <b>c</b> : lower limit of the middle variable <b>d</b> : upper limit of the middle variable <b>w</b> : outer variable of integration <b>r</b> : lower limit of the outer variable <b>s</b> : upper limit of the outer variable
<b>Output</b>	the definite integral $\int_r^s \int_c^d \int_a^b f(u, v, w) \, dudvdw$
<b>Examples</b>	<b>m.int3</b> (( $\rho \sin(\phi) \sin(\theta)$ ) <sup>2</sup> · $\rho \cdot \rho^2 \sin(\phi)$ , $\rho, 0, 2, \phi, 0, \pi, \theta, -\frac{\pi}{2}, \frac{\pi}{2}$ ) returns $\int_{-\pi/2}^{\pi/2} \int_0^{\pi} \int_0^2 (\rho \sin(\phi) \sin(\theta))^2 \cdot \rho \cdot \rho^2 \sin(\phi) \, d\rho d\phi d\theta = \frac{64\pi}{9} \approx 22.34$ . $\frac{1}{24} \cdot \mathbf{m.int3}([x, y, z], x, 0, 2, z, 0, 4, y, 0, 3)$ returns $\left[ 1, \frac{3}{2}, 2 \right]$ .
<b>See Also</b>	<b>INT1</b> , <b>INT2</b> (for inline single and double integrals, respectively)
<b>Note</b>	As the examples show, it is unnecessary for the variables of integration to be in alphabetical order.

---

---

## INTG

<b>Description</b>	The <b>intg</b> function computes an indefinite integral (antiderivative) in an inline fashion.
<b>Access</b>	<b>a.intg</b> library shortcut in the <i>auxiliary</i> group
<b>Syntax</b>	<b>a.intg(f,v)</b>
<b>Input</b>	<b>f</b> : a scalar, vector, or matrix expression expression in <b>v</b> <b>v</b> : variable of integration
<b>Output</b>	the indefinite integral (antiderivative) $\int f dx$
<b>Examples</b>	<b>a.intg</b> ( $x^2, x$ ) returns $\int x^2 dx = \frac{1}{3}x^3$ . <b>a.intg</b> ( $[x, x^2, x^3], x$ ) returns $\int [x, x^2, x^3] dx = [\frac{1}{2}x^2, \frac{1}{3}x^3, \frac{1}{4}x^4]$ .
<b>See Also</b>	<b>INT1</b> (to compute an inline definite integral, $\int_a^b f dx$ )
<b>Note</b>	If $f$ is a scalar expression in $x$ , then <b>a.intg</b> ( $f, [x, C]$ ) returns $\int f dx + C$ , the most general antiderivative of $f$ . This could also be realized via <b>a.intg</b> ( $f, x$ ) + <b>C</b> instead.

---

## INVFUNC

<b>Description</b>	The <b>invfunc</b> function computes the inverse of a function $y = f(x)$ .
<b>Access</b>	<b>f.invfunc</b> library shortcut in the <i>functions</i> group
<b>Syntax</b>	<b>f.invfunc(eq,x)</b>
<b>Input</b>	<b>eq</b> : an equation expressing a dependent variable explicitly in terms of an independent variable <b>x</b> : the independent variable in the equation
<b>Output</b>	the inverse functional expression $f^{-1}(x)$ , if successful
<b>Example</b>	<b>f.invfunc</b> ( $y = (2 + 5x)^{1/3}, x$ ) returns $\frac{x^3 - 2}{5}$ .
<b>See Also</b>	<b>SOLVE</b> in TI-Nspire CAS documentation

---

## JMATRIX

<b>Description</b>	The <b>jmatrix</b> function computes the Jacobian matrix of a transformation.
<b>Access</b>	<b>d.jmatrix</b> library shortcut in the <i>derivatives</i> group
<b>Syntax</b>	<b>d.jmatrix(T,v)</b>
<b>Input</b>	<b>T</b> : a vector of expressions giving old variables $x_1, \dots, x_n$ in terms of new ones $u_1, \dots, u_n$ ; i.e., $[x_1(u_1, \dots, u_n), \dots, x_n(u_1, \dots, u_n)]$ <b>v</b> : a vector of new variables, $[u_1, \dots, u_n]$
<b>Output</b>	The Jacobian matrix $J = \begin{bmatrix} \partial x_1 / \partial u_1 & \cdots & \partial x_1 / \partial u_n \\ \vdots & \ddots & \vdots \\ \partial x_n / \partial u_1 & \cdots & \partial x_n / \partial u_n \end{bmatrix}$
<b>Example</b>	<b>d.jmatrix</b> ( $[r \cdot \cos(\theta), r \cdot \sin(\theta)], [r, \theta]$ ) returns $\begin{bmatrix} \cos \theta & -r \sin \theta \\ \sin \theta & r \cos \theta \end{bmatrix}$ .
<b>See Also</b>	<b>DET</b> in TI-Nspire CAS documentation

---

---

## LAPL

<b>Description</b>	The <b>lapl</b> function computes the Laplacian of a scalar field.
<b>Access</b>	<b>d.lapl</b> library shortcut in the <i>derivatives</i> group
<b>Syntax</b>	<b>d.lapl(<i>f</i>,<i>v</i>)</b>
<b>Input</b>	<b>f</b> : a scalar field <b>v</b> : a vector of (typically two or three) independent variables in which <b>f</b> is expressed
<b>Output</b>	the Laplacian $\nabla^2 f = \sum_{k=1}^n \frac{\partial^2 f}{\partial x_k^2}$ of the scalar field <i>f</i> ; typically, $\frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$ or $\frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} + \frac{\partial^2 f}{\partial z^2}$
<b>Example</b>	<b>d.lapl</b> (sin( <i>x</i> ) cosh( <i>y</i> ) + cos( <i>x</i> ) sinh( <i>y</i> ), [ <i>x</i> , <i>y</i> ]) returns 0.
<b>See Also</b>	<b>CURL, DIV</b>

---

## LINPTDIR

<b>Description</b>	The <b>linptdir</b> function finds a parametric representation of the (hyper)line through a point in the direction of a vector.
<b>Access</b>	<b>g.linptdir</b> library shortcut in the <i>geometry</i> group
<b>Syntax</b>	<b>g.linptdir(<i>a</i>,<i>v</i>)</b>
<b>Input</b>	<b>a</b> : a point on the (hyper)line as a position vector <b>v</b> : a direction vector having the same number of components as <b>a</b>
<b>Output</b>	a parametric representation $\mathbf{L}(t) = \mathbf{a} + t\mathbf{v}$ of the (hyper)line
<b>Example</b>	<b>g.linptdir</b> ([8, 1, 6], [2, -5, 1]) returns [2 <i>t</i> + 8, 1 - 5 <i>t</i> , <i>t</i> + 6].
<b>See Also</b>	<b>CLINE2PT, PLINE2PT</b>

---

---

## LIS

**Description** The **lis** function computes the line integral of a scalar field along a curve  $C$  with respect to arc length.

**Access** **i.lis** library shortcut in the *integrals* group

**Syntax** **i.lis( $f, \mathbf{v}, \mathbf{g}, t, \mathbf{a}, \mathbf{b}$ )**

**Input**  
 **$f$** : an expression representing the scalar field  
 **$\mathbf{v}$** : a row vector (of two or more elements): the variables in which  **$f$**  is expressed  
 **$\mathbf{g}$** : a row vector that represents the parameterization of the curve  $C$   
 **$t$** : the variable of integration (typically  $t$ )  
 **$\mathbf{a}$** : lower bound of the variable of integration in the line integral  
 **$\mathbf{b}$** : upper bound of the variable of integration in the line integral

**Output** the line integral  $\int_C f ds = \int_a^b f(\mathbf{g}(t)) \|\mathbf{g}'(t)\| dt$

**Example**  $7 + x^2 + y^4 \rightarrow \mathbf{f}: [2 \cos(t), 2 \sin(t)] \rightarrow \mathbf{g}: \mathbf{i.lis}(\mathbf{f}, [x, y], \mathbf{g}, t, 0, \pi/4)$  returns  $\frac{3}{2}(5\pi - 4) \approx 17.56$ .

**See Also** **LIV**

---

## LIV

**Description** The **liv** function computes the line integral of a vector field along a curve  $C$ .

**Access** **i.liv** library shortcut in the *integrals* group

**Syntax** **i.liv( $\mathbf{w}, \mathbf{v}, \mathbf{g}, t, \mathbf{a}, \mathbf{b}$ )**

**Input**  
 **$\mathbf{w}$** : a row vector representing the vector field  
 **$\mathbf{v}$** : a row vector (of two or more elements): the variables in which  **$\mathbf{w}$**  is expressed  
 **$\mathbf{g}$** : a row vector that represents the parameterization of the curve  $C$   
 **$t$** : the variable of integration (typically  $t$ )  
 **$\mathbf{a}$** : lower bound of the variable of integration in the line integral  
 **$\mathbf{b}$** : upper bound of the variable of integration in the line integral

**Output** the line integral  $\int_C \mathbf{w} \cdot d\mathbf{g} = \int_a^b \mathbf{w}(\mathbf{g}(t)) \cdot \mathbf{g}'(t) dt$

**Example**  $[x \cdot y, 4z - x, x^2 z] \rightarrow \mathbf{w}: [3t^2 - 8, 4t^3, 6t + 7] \rightarrow \mathbf{g}: \mathbf{i.liv}(\mathbf{w}, [x, y, z], \mathbf{g}, t, -1, 1)$  returns  $\frac{30804}{7} \approx 4400.57$ .

**See Also** **LIS**

---

---

## LPMD

<b>Description</b>	The <b>LPMD</b> function computes the leading principal minor determinants of a Hessian matrix.
<b>Access</b>	<b>f.lpmd</b> library shortcut in the <i>functions</i> group
<b>Syntax</b>	<b>f.lpmd(<i>H</i>)</b>
<b>Input</b>	<b><i>H</i></b> : a symbolic or numerical Hessian matrix
<b>Output</b>	a list of the leading principal minor determinants of the input matrix <b><i>H</i></b> ; i.e., the determinants of the upper left square submatrices of <b><i>H</i></b> : $ H_{11} $ , $\begin{vmatrix} H_{11} & H_{12} \\ H_{21} & H_{22} \end{vmatrix}$ , etc.
<b>Example</b>	<b>f.hess</b> ( $x^3 + 4x \cdot y - y^2$ , $[x, y]$ ) returns $\begin{bmatrix} 6x & 4 \\ 4 & -2 \end{bmatrix}$ . Then <b>f.lpmd(Ans)</b> yields $\{6x, -12x - 16\}$ .
<b>See Also</b>	<b>HESS</b>

---

## ORIENT

<b>Description</b>	The <b>orient</b> function computes a normal vector that is perpendicular to an orientable surface.
<b>Access</b>	<b>i.orient</b> library shortcut in the <i>integrals</i> group
<b>Syntax</b>	<b>i.orient(<i>s, u, v</i>)</b>
<b>Input</b>	<b><i>s</i></b> : a row vector expression in <b><i>u</i></b> and <b><i>v</i></b> having three components that represent the <i>x</i> -, <i>y</i> -, and <i>z</i> -coordinates of an orientable parametric surface <b><i>u</i></b> : one of the parameters in the surface representation <b><i>v</i></b> : the other parameter in the surface representation
<b>Output</b>	the cross product $\frac{\partial \mathbf{s}}{\partial u} \times \frac{\partial \mathbf{s}}{\partial v}$ , which is perpendicular to the orientable surface
<b>Examples</b>	$[x, y, x^2 + y^2] \rightarrow$ <b>s: i.orient(<i>s, x, y</i>)</b> returns $[-2x, -2y, 1]$ , signifying an upward pointing normal vector to the surface
<b>See Also</b>	<b>SIV</b>

---

## ORTH

<b>Description</b>	The <b>orth</b> function computes the orthogonal projection of one vector onto another.
<b>Access</b>	<b>v.orth</b> library shortcut in the <i>vectors</i> group
<b>Syntax</b>	<b>v.orth(<i>v, w</i>)</b>
<b>Input</b>	<b><i>v</i></b> : a vector having two or more components <b><i>w</i></b> : a vector with the same number of components as <b><i>v</i></b>
<b>Output</b>	the orthogonal projection $\text{orth}_v \mathbf{w} = \mathbf{w} - \text{proj}_v \mathbf{w}$ of the vector <b><i>w</i></b> onto the vector <b><i>v</i></b> , where $\text{proj}_v \mathbf{w}$ is the parallel projection of <b><i>w</i></b> onto <b><i>v</i></b>
<b>Example</b>	<b>v.orth</b> ( $[6, 1, -2]$ , $[0, 4, 7]$ ) returns $\left[\frac{60}{41}, \frac{174}{41}, \frac{267}{41}\right] \approx [1.46, 4.24, 6.51]$ .
<b>See Also</b>	<b>COMP, PROJ</b>

---

---

## PERP

<b>Description</b>	The <b>perp</b> function computes an orthogonal complement of a two-dimensional vector.
<b>Access</b>	<b>v.perp</b> library shortcut in the <i>vectors</i> group
<b>Syntax</b>	<b>v.perp(v)</b>
<b>Input</b>	<b>v</b> : a vector having two components
<b>Output</b>	an orthogonal complement $\mathbf{v}^\perp = [-v_2, v_1]$ of the vector $\mathbf{v} = [v_1, v_2]$
<b>Example</b>	<b>v.perp</b> ([6, 1]) returns [-1, 6].
<b>See Also</b>	<b>ORTH</b>

---

## PLANE3PT

<b>Description</b>	The <b>plane3pt</b> function gives Cartesian representations of the plane through three points in space.
<b>Access</b>	<b>g.plane3pt</b> library shortcut in the <i>geometry</i> group
<b>Syntax</b>	<b>g.plane3pt(u, v, w)</b>
<b>Input</b>	<b>u</b> : a point on the plane as a position vector <b>v</b> : a different point on the plane as a position vector <b>w</b> : a third point on the plane as a position vector; <b>u</b> , <b>v</b> , and <b>w</b> are noncollinear
<b>Output</b>	Cartesian representations of the plane; variations of $\mathbf{n} \cdot [x, y, z] = \mathbf{n} \cdot \mathbf{u}$ , where $\mathbf{n} = \overrightarrow{uv} \times \overrightarrow{uw} = (\mathbf{v} - \mathbf{u}) \times (\mathbf{w} - \mathbf{u})$
<b>Example</b>	<b>g.plane3pt</b> ([0, 4, 6], [5, 1, -1], [2, 6, 0]) returns the matrix $\begin{bmatrix} x = 5 - \frac{1}{2}y - \frac{1}{2}z & y = 10 - 2x - z \\ z = 2x - y + 10 & 32x + 16y + 16z = 160 \end{bmatrix}$ , useful in setting up ranges of integration in triple integrals.
<b>See Also</b>	<b>PLANEINT</b> , <b>PLANENPT</b>

---

## PLANEINT

<b>Description</b>	The <b>planeint</b> function gives Cartesian representations of the plane in space through specified axis intercepts.
<b>Access</b>	<b>g.planeint</b> library shortcut in the <i>geometry</i> group
<b>Syntax</b>	<b>g.planeint(a, b, c)</b>
<b>Input</b>	<b>a</b> : the <i>x</i> -intercept of the plane <b>b</b> : the <i>y</i> -intercept of the plane <b>c</b> : the <i>z</i> -intercept of the plane
<b>Output</b>	Cartesian representations of the plane; variations of $\frac{x}{a} + \frac{y}{b} + \frac{z}{c} = 1$
<b>Example</b>	<b>g.planeint</b> (2, 3, 6) returns the matrix $\begin{bmatrix} x = 2 - \frac{2}{3}y - \frac{1}{3}z & y = 3 - \frac{3}{2}x - \frac{1}{2}z \\ z = 6 - 3x - 2y & \frac{x}{2} + \frac{y}{3} + \frac{z}{6} = 1 \end{bmatrix}$ , useful in setting up ranges of integration in triple integrals.
<b>See Also</b>	<b>PLANE3PT</b> , <b>PLANENPT</b>

---

---

## PLANENPT

<b>Description</b>	The <b>planenpt</b> function gives Cartesian representations of the plane through a point having a given normal vector.
<b>Access</b>	<b>g.planenpt</b> library shortcut in the <i>geometry</i> group
<b>Syntax</b>	<b>g.planenpt(<i>n</i>,<i>p</i>)</b>
<b>Input</b>	<b><i>n</i></b> : a normal vector perpendicular to the plane <b><i>p</i></b> : a point on the plane as a position vector
<b>Output</b>	Cartesian representations of the plane; variations of $\mathbf{n} \cdot [x, y, z] = \mathbf{n} \cdot \mathbf{p}$
<b>Example</b>	<b>g.planenpt</b> ([3, -4, -7], [8, 2, 0]) returns the matrix $\begin{bmatrix} x = \frac{16}{3} + \frac{4}{3}y + \frac{7}{3}z & y = \frac{3}{4}x - \frac{7}{4}z - 4 \\ z = \frac{3}{7}x - \frac{4}{7}y - \frac{16}{7} & 3x - 4y - 7z = 16 \end{bmatrix}$ , useful in setting up ranges of integration in triple integrals.
<b>See Also</b>	<b>PLANEINT, PLANE3PT</b>

---

## PLINE2PT

<b>Description</b>	The <b>pline2pt</b> function finds a parametric representation of the (hyper)line between two points.
<b>Access</b>	<b>g.pline2pt</b> library shortcut in the <i>geometry</i> group
<b>Syntax</b>	<b>g.pline2pt(<i>a</i>,<i>b</i>)</b>
<b>Input</b>	<b><i>a</i></b> : a point on the (hyper)line as a position vector <b><i>b</i></b> : a different point on the (hyper)line as a position vector
<b>Output</b>	a parametric representation $\mathbf{L}(t) = \mathbf{a} + t(\mathbf{b} - \mathbf{a})$ of the (hyper)line
<b>Example</b>	<b>g.pline2pt</b> ([-3, 5, -2], [-4, 6, 7]) returns $[-t - 3, t + 5, 9t - 2]$ .
<b>See Also</b>	<b>CLINE2PT, LINPTDIR</b>

---

## POT

<b>Description</b>	The <b>pot</b> function determines whether a vector field is conservative. If so, it returns a potential function of the vector field; if not, <b>false</b> is returned.
<b>Access</b>	<b>d.pot</b> library shortcut in the <i>derivatives</i> group
<b>Syntax</b>	<b>d.pot(<i>w</i>,<i>v</i>)</b>
<b>Input</b>	<b><i>w</i></b> : a vector field <b><i>v</i></b> : a row vector of independent variables with the same number of components as <b><i>w</i></b>
<b>Output</b>	a potential function $f$ for <b><i>w</i></b> (i.e., $\mathbf{w} = \vec{\nabla}f$ ) if <b><i>w</i></b> is conservative; otherwise, <b>false</b>
<b>Example</b>	<b>d.pot</b> ([ $4xe^z$ , $\cos(y)$ , $2x^2e^z$ ], [x, y, z]) returns $2x^2e^z + \sin y$ .
<b>See Also</b>	<b>FTLI</b>

---

---

## PROJ

<b>Description</b>	The <b>proj</b> function computes the parallel projection of one vector onto another.
<b>Access</b>	<b>v.proj</b> library shortcut in the <i>vectors</i> group
<b>Syntax</b>	<b>v.proj(v,w)</b>
<b>Input</b>	<b>v</b> : a vector with two or more components <b>w</b> : a vector having the same number of components as <b>v</b>
<b>Output</b>	the parallel projection $\text{proj}_{\mathbf{v}}\mathbf{w} = \left(\frac{\mathbf{v} \cdot \mathbf{w}}{\ \mathbf{v}\ ^2}\right) \frac{\mathbf{v}}{\ \mathbf{v}\ }$ of the vector <b>w</b> onto the vector <b>v</b>
<b>Example</b>	<b>v.proj</b> ([6, 1, -2], [0, 4, 7]) returns $\left[-\frac{60}{41}, -\frac{10}{41}, \frac{20}{41}\right] \approx [-1.46, -0.24, 0.49]$ .
<b>See Also</b>	<b>COMP, ORTH</b>

---

## SIS

<b>Description</b>	The <b>sis</b> function computes the surface integral of a scalar field.
<b>Access</b>	<b>i.sis</b> library shortcut in the <i>integrals</i> group
<b>Syntax</b>	<b>i.sis(f,s, u,g,h, v,a,b)</b>
<b>Input</b>	<b>f</b> : an expression that represents the scalar field <b>s</b> : a row vector giving a parameterization of the surface in terms of the variables of integration <b>u</b> : inner variable of integration <b>g</b> : lower limit of the inner variable <b>h</b> : upper limit of the inner variable <b>v</b> : outer variable of integration <b>a</b> : lower limit of the outer variable <b>b</b> : upper limit of the outer variable
<b>Output</b>	the surface integral $\iint_S f dS = \int_a^b \int_g^h f(\mathbf{s}(u,v)) \ \mathbf{s}_u \times \mathbf{s}_v\  du dv$
<b>Example</b>	$7 + x^2 + y^2 \rightarrow \mathbf{f}: [r \cdot \cos(\theta), r \cdot \sin(\theta), 36 - 4r^2] \rightarrow \mathbf{s}:$ <b>i.sis</b> ( <b>f, s, r,0,3, θ,0,2π</b> ) returns $\frac{(381397\sqrt{577} - 373)\pi}{5120} \approx 5621.18$ .
<b>See Also</b>	<b>SIV</b>

---

---

## SIV

<b>Description</b>	The <b>siv</b> function computes the surface integral of a vector field.
<b>Access</b>	<b>i.siv</b> library shortcut in the <i>integrals</i> group
<b>Syntax</b>	<b>i.siv(w,s, u,g,h, v,a,b)</b>
<b>Input</b>	<b>w</b> : an expression that represents the vector field <b>s</b> : a row vector giving a parameterization of the surface in terms of the variables of integration <b>u</b> : inner variable of integration <b>g</b> : lower limit of the inner range of integration <b>h</b> : upper limit of the inner range of integration <b>v</b> : outer variable of integration <b>a</b> : lower limit of the outer range of integration <b>b</b> : upper limit of the outer range of integration
<b>Output</b>	The surface integral $\iint_S \mathbf{w} \cdot d\mathbf{S} = \int_a^b \int_g^h \mathbf{w}(\mathbf{s}(u,v)) \cdot (\mathbf{s}_u \times \mathbf{s}_v) \, du \, dv$
<b>Example</b>	$[e^y, ye^x, x^2y] \rightarrow \mathbf{w}: [x, y, x^2 + y^2] \rightarrow \mathbf{s}:$ <b>i.siv(w, s, x, 0, 1, y, 0, 1)</b> returns $\frac{11}{6} - \frac{5e}{3} \approx -2.70$ .
<b>See Also</b>	<b>ORIENT, SIS</b>
<b>Note</b>	If the parameterization of $\mathbf{s}(u,v)$ is such that the normal to the surface, $\mathbf{s}_u \times \mathbf{s}_v$ , has the opposite orientation to the one desired, simply negate the <b>siv</b> command. In other words, specify <b>-i.siv(...)</b> . The normal vector is determined by the order of the variables of integration in the command. In the example, the normal vector for the parameterization is $\mathbf{s}_x \times \mathbf{s}_y = [-2x, -2y, 1]$ , which gives an upward pointing normal since the <b>k</b> -component is positive. This may be quickly computed by using <b>orient</b> , <i>q.v.</i>

---

## STP

<b>Description</b>	The <b>stp</b> function computes the scalar triple product of three vectors.
<b>Access</b>	<b>v.stp</b> library shortcut in the <i>vectors</i> group
<b>Syntax</b>	<b>v.stp(a,b,c)</b>
<b>Input</b>	<b>a</b> : a vector with 3 components <b>b</b> : another vector with 3 components <b>c</b> : a third vector with 3 components
<b>Output</b>	the scalar triple product, $\mathbf{a} \cdot (\mathbf{b} \times \mathbf{c})$
<b>Example</b>	<b>v.stp([3, -2, 7], [-4, 0, 5], [-6, 9, 8])</b> returns $-391$ .
<b>See Also</b>	<b>DOTP, CROSSP</b> in TI-Nspire CAS documentation

---

## SUM1

**Description** The **sum1** function computes approximations to the single integral  $\int_a^b f(x) dx$  using five different methods: left endpoint approximation, right endpoint approximation, Trapezoidal Rule, Midpoint Rule, and Simpson's Rule.

**Access** **m.sum1** library shortcut in the *multiple* group

**Syntax** **m.sum1(f,x,a,b,n)**

**Input** **f**: a formulaic expression in the case of continuous data or a vector of function values in the case of discrete data  
**a**: lower limit of integration  
**b**: upper limit of integration  
**n**: number of subintervals used in each approximation

**Output** Numerical approximations to  $\int_a^b f(x) dx$  are returned. Here are the formulas for the five methods. In each  $h = \frac{b-a}{n}$ .

left	$h \sum_{k=0}^{n-1} f(a+kh)$
right	$h \sum_{k=1}^n f(a+kh)$
trap	$h \left( \frac{f(a)+f(b)}{2} + \sum_{k=1}^{n-1} f(a+kh) \right)$
mid	$h \sum_{k=0}^{n-1} f \left( a + \left( k + \frac{1}{2} \right) h \right)$
simp	$\frac{h}{3} \left( f(a) + f(b) + 4 \sum_{j=1}^{n/2} f(a+(2j-1)h) + 2 \sum_{k=1}^{n/2-1} f(a+2kh) \right)$

**Examples** Consider  $\int_2^3 \frac{1}{x} dx = \ln \frac{3}{2} \approx 0.405465$ . The command **m.sum1**( $\frac{1}{x}$ , x, 2, 3, 10) returns the following approximations.

left	right	trap	mid	simp
0.413914	0.397247	0.405581	0.405407	0.405465

Similarly, consider  $\int_0^4 f(x) dx$  where **f** is given by discrete values in a vector as follows.

**DelVar f: [0, 1, 3, 4.5, 5, 4.5, 3, 2, 1]→f**

Then the command **m.sum1**(**f**, x, 0, 4, **n**) returns the following approximations. The routine in this case automatically computes **n** to be one fewer than the number of elements in the vector **f**; in other words, the number of equal-length subintervals of  $[a, b]$  determined by the data. For the Midpoint Rule,  $n/2$  subintervals are used.

left	right	trap	mid	simp
11.5	12	11.75	12	11.833333

**See Also**  $\approx$  (Control-Enter) in the case of a formulaic expression for **f** to compute a very accurate approximation to the integral

- Notes**
- The number of subintervals **n** must be a positive *even* integer in the case of Simpson's Rule and for the Midpoint Rule when discrete data is specified.
  - If **f** is given by a formula, then **n** may be symbolic (except for Simpson's Rule).

---

## SUM2

**Description** The **sum2** function computes approximations to the double integral  $\int_c^d \int_a^b f(x,y) dx dy$  using five different methods. Function values are respectively sampled at the following points of each subrectangle of the partition of integration: top left, top right, middle (center), bottom left, and bottom right.

TL	◆	TR
◆	MD	◆
BL	◆	BR

**Access** **m.sum2** library shortcut in the *multiple* group

**Syntax** **m.sum2(f, x,a,b,m, y,c,d,n)**

**Input** **f**: a scalar expression in the variables of integration  
**x**: inner variable of integration  
**a**: lower limit of inner variable  
**b**: upper limit of inner variable  
**m**: number of subintervals for the inner variable  
**y**: outer variable of integration  
**a**: lower limit of outer variable  
**b**: upper limit of outer variable  
**n**: number of subintervals for the outer variable

**Output** Numerical approximations to  $\int_c^d \int_a^b f(x,y) dx dy$  are returned, each having the form  $hk \sum_{i=1}^m \sum_{j=1}^n f(x_i, y_j)$ . Here  $h = \frac{b-a}{m}$  and  $k = \frac{d-c}{n}$  with  $(x_i, y_j)$  being one of the five aforementioned points in subrectangles of the partition.

**Examples** Consider  $\int_{3/4}^1 \int_0^{1/2} e^{x+y} dx dy = e^{3/2} - e^{5/4} - e + e^{3/4} \approx 0.390064$ . The command **m.sum2**( $e^{x+y}, x, 0, \frac{1}{2}, 4, y, \frac{3}{4}, 1, 8$ ) returns the following approximations.

0.371945	◆	0.421468
◆	0.389795	◆
0.360501	◆	0.408501

**See Also**  $\approx$  (Control-Enter) to rapidly compute a very accurate approximation to the integral

**Note** Make certain that the variables of integration are in alphabetical order! That is, the inner variable should come before the outer variable in the alphabet(s). Be apprised that Roman letters come before Greek letters in the collating sequence order. Hence put *r* before  $\theta$ . Similarly, put  $\theta$  before  $\phi$ .

---

---

## SUM3

**Description** The **sum3** function computes an approximation to the triple integral  $\int_r^s \int_c^d \int_a^b f(x,y,z) dx dy dz$  using the middle (center) points of each subbox in the partition for function evaluation.

**Access** **m.sum3** library shortcut in the *multiple* group

**Syntax** **m.sum3(f, x,a,b,l, y,c,d,m, z,r,s,n)**

**Input** **f**: a scalar expression in the variables of integration  
**x**: inner variable of integration  
**a**: lower limit of inner variable  
**b**: upper limit of inner variable  
**l**: number of subintervals for the inner variable  
**y**: middle variable of integration  
**a**: lower limit of middle variable  
**b**: upper limit of middle variable  
**m**: number of subintervals for the middle variable  
**z**: outer variable of integration  
**r**: lower limit of outer variable  
**s**: upper limit of outer variable  
**n**: number of subintervals for the outer variable

**Output** The approximation returned is

$$\sum_{i=1}^l \sum_{j=1}^m \sum_{k=1}^n f(x_i, y_j, z_k) \Delta x \Delta y \Delta z, \text{ where}$$
$$x_i = a + \left(i - \frac{1}{2}\right) h_1 \text{ and } h_1 = \Delta x = \frac{b-a}{l},$$
$$y_j = c + \left(j - \frac{1}{2}\right) h_2 \text{ and } h_2 = \Delta y = \frac{d-c}{m},$$
$$z_k = r + \left(k - \frac{1}{2}\right) h_3 \text{ and } h_3 = \Delta z = \frac{s-r}{n}.$$

**Example** Consider  $\int_0^1 \int_0^1 \int_0^1 e^{x^3+y^3+z^3} dx dy dz \approx 2.416377$ . The command **m.sum3**( $e^{x^3+y^3+z^3}$ ,  $x, 0, 1, 4$ ,  $y, 0, 1, 4$ ,  $z, 0, 1, 4$ ) returns 2.308752.

**See Also**  $\approx$  (Control-Enter) to rapidly compute a very accurate approximation to the integral

**Note** Make certain that the variables of integration are in alphabetical order! That is, the inner variable should come before the outer variable in the alphabet(s). Be apprised that Roman letters come before Greek letters in the collating sequence order. Hence put  $r$  before  $z$  before  $\theta$ . Similarly, put  $\theta$  before  $\phi$  before  $\rho$ .

---

## SUMV

**Description** The **sumv** function computes the sum of the elements of a vector.

**Access** **a.sumv** library shortcut in the *auxiliary* group

**Syntax** **a.sumv(v)**

**Input** **v**: a vector with 2 or more elements

**Output** the sum of the elements of **v**

**Example** **a.sumv**([1, 2, 3]) returns 6.

**See Also** **SUM** in TI-Nspire CAS documentation.

---

## U2U

**Description** The **u2u** program converts an expression in  $t$  to one in  $x$  for graphing purposes.

**Access** **a.u2u** library shortcut in the *auxiliary* group

**Syntax** **a.u2u(e, "f")**

**Input** **e**: a scalar or vector expression in  $t$ , possibly involving Heaviside unit step functions  
**"f"**: the name of the function (a string) which will contain the converted expression

**Output** The expression **e** has all occurrences of the variable  $t$  changed to  $x$  and function references to  $u()$  changed to  $\hat{u}()$ . The command then defines the Heaviside unit step function  $\hat{u}(t) = \begin{cases} 1, & t \geq 0 \\ 0, & t < 0 \end{cases}$ .

**Examples** In differential equations, it is seen that the solution to the initial value problem

$$y' + 5y = 2u(t - 3), \quad y(0) = 1$$

where  $u$  is the Heaviside unit step function, is given by

$$e^{-5t} \left( u(t) - \frac{2e^{15}u(t-3)}{5} \right) + \frac{2u(t-3)}{5}.$$

Then **a.u2u(Ans, "f")** defines the function

$$f(x) = e^{-5x} \left( \hat{u}(x) - \frac{2e^{15}\hat{u}(x-3)}{5} \right) + \frac{2\hat{u}(x-3)}{5}$$

where  $\hat{u}(t) = \begin{cases} 1, & t \geq 0 \\ 0, & t < 0 \end{cases}$ . One may then use the Graphs and Geometry application to graph the solution by referencing  $f(x)$ .

**See Also** | (substitution facility) in TI-Nspire CAS documentation

---