Graphing / Equations

|  |  |  |
| --- | --- | --- |
| **Program** | **Description** | **Usage** |
| sketch(graph,  x\_min, x\_max,  y\_min, y\_max)  **For decimal answers use:**  sketchD(graph,  x\_min, x\_max,  y\_min, y\_max) | Graphs the supplied expression in the domain and range, includes:   * vertical asymptotes * horizontal asymptotes * axis-intercepts * turning points * points of inflexion | sketch((x^2-2x)/(x^2-2x-3), -4, 4, -6, 6)  **will:**   * view the window to x: [-4,4] and y: [-6, 6] * draw (x^2-2x)/(x^2-2x-3) * plot asymptotes in dotted line * label coordinates * list information in textual format beneath:   + Start = start point   + End = end point   + V-asym   + H-asym   + X-ints   + Y-int   + TP = turning point   + POI = non-stationary point in inflexion   + SPOI = stationary point in inflexion * Tap the graph to remove the info if it becomes too cluttered |
| sketchA(graph,  x\_min, x\_max)  **For decimal answers use:**  sketchAD | Like sketch(…) above but will automatically find a suitable range | sketchA(x+2,-1,1) |
| axisints(equation) | Returns the coordinates of all axial intercepts of the supplied equation | axisints(y=x+3)   * {{-3,0},{0,3}} |
| ellipse(equation) | Finds the center coordinates, the horizontal radius and the vertical radius from the supplied ellipse equation | ellipse()  \* Results returned in a matrix format |
| hyperbol(equation) | Finds the center coordinates and the asymptote equations from the supplied hyperbola equation | hyperbol ()  \* Results returned in a matrix format |
| inv(equation) | Returns the inverse relation of the supplied equation | inv(y=x^2)   * , |
| tToCar(x\_equation,y\_equation) | Converts the parametric equations in terms of ‘t’ to the equivalent Cartesian equation in terms of ‘x’ | tToCar(x=t-1, y=t^2+t) |

Vectors

|  |  |  |
| --- | --- | --- |
| **Program** | **Description** | **Usage** |
| linDepen(list\_of\_vectors) | Returns the solutions of the coefficients for linear dependence. “No Solution” is returned if the vectors are linearly independent. | linDepen()   * , |
| scaRes(a,b) | Returns the scalar resolute of vectors a and b | scaRes() |
| vecRes(a,b) | Returns the vector resolute of a in direction of b | vecRes() |

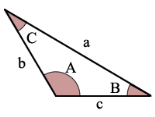
Complex Numbers

|  |  |  |
| --- | --- | --- |
| **Program** | **Description** | **Usage** |
| Argument(arg) | Returns the equivalent Argument angle (-π, π] from any argument. | Argument(7 π/4) |
| cis(angle) | Shortcut for: cos(angle)+isin(angle) | cis(π/3) |
| compRoot(power, complex\_number) | Solves complex roots in the form zx = y using deMoivre’s theorem | compRoot (2,) |
| zToCar(complex\_relation)  **Must be in ‘Cplx’ mode** | Converts the supplied complex relation in terms of z into the equivalent Cartesian equation(s) in terms of x. | zToCar(|z+2|=|z-2|) |

Sine and Cos Rules

|  |  |  |
| --- | --- | --- |
| **Program** | **Description** | **Usage** |
| cosRuleA(a,b,c) | Finds the angle A from lengths a,b,c | cosRuleA(7,3,5) |
| cosRulea(A,b,c) | Finds the length a from angle A and lengths b,c | cosRulea(π/3,3,5) |
| sinRuleA(a,b,B) | Finds the possible angle(s) A from length a and b and angle B. | sinRuleA(√3,1, π/6) |
| sineRulea(A,b,B) | Finds the length of a from angles A and B and length b | sineRulea(π/3,2, π/6) |

# ! Convention

The return values and parameters match the diagram below: 

Circular Functions

|  |  |  |
| --- | --- | --- |
| **Program** | **Description** | **Usage** |
| cosec(θ) | Shortcut for:  1/sin(θ) | cosec(π/3) |
| sec(θ) | Shortcut for:  1/cos(θ) | sec(π/3) |
| cot(θ) | Shortcut for:  1/tan(θ) | cot(π/3) |

Calculus

|  |  |  |
| --- | --- | --- |
| **Program** | **Description** | **Usage** |
| invDiff(function\_terms\_of\_y) | Finds the gradient function(s) in terms of **x** for the supplied function in terms of **y** | invDiff(y^2-2y)  invDiff(y^2-2y)|x=0 (find gradient(s) at x=0) |
| intSub(integral,start\_x,end\_x,  u\_substitution) | Performs integral substitution on the supplied expression, returning it in terms of u. | intSub(, 0, 2, )  Which should be interpreted as |
| area(func,start\_x,end\_x) | Finds the area bounded by:   * graph ‘func’ * x-axis * x=start\_x * x=end\_x | area(x^2,-2,2) |
| aready(func,start\_x,end\_x) | Finds the area bounded by   * graph ‘func’ * **y-axis** * y=func(start\_x) * y=func(end\_x) | aready(x^2,0,2) |
| euler(dydx)  **or**  eulerx(dydx) | Performs euler’s method of linear approximation on the supplied dy/dx function | euler(x^2-2x) and follow the prompts to enter x0, y0, h and # iterations  Returns the values for the final iteration  **Or**  eulerx(x^2-2x) and follow the prompts to enter x0, y0, h and the x-value of the final iteration |

Miscellaneous

|  |  |  |
| --- | --- | --- |
| **Program** | **Description** | **Usage** |
| isInf(value) | Returns whether the supplied value is positive infinity | isInf(999)   * FALSE |
| isUndef(value) | Returns whether the supplied value is undefined | isUndef (1/0)   * TRUE |
| istypeof(value, type) | Returns whether the supplied value contains a certain type of value: | istypeof({1,2}, “LIST”)   * TRUE |
| matToEq( join\_operator\_str, variables, coefficient\_matrices, result\_matrix) | Best explained through example | matToEq(“+”, ) |
| dCoord(x\_value, y\_value, label) | Labels the coordinate on the active graph. Returns the labelled coordinate string | dCoord(2,4, “TP”)   * “TP(2,4)” |
| dHAsymp(y\_value, line\_pixel\_interval) | Draws a horizontal asymptote line on the active graph. The line is dotted with an active pixel every ‘line\_pixel\_interval’ pixels | dHAsymp(4/3, 4) |
| dVAsymp(x\_value, line\_pixel\_interval) | Draws a vertical asymptote line on the active graph. The line is dotted with an active pixel every ‘line\_pixel\_interval’ pixels | dVAsymp(-2, 3) |
| xToPxl(x\_value) | Returns the nearest corresponding x-direction pixel for the active graph | xToPxl(0)   * 77 |
| yToPxl(y\_value) | Returns the nearest corresponding y-direction pixel for the active graph | yToPxl(0)   * 38 |
| strRep(search,replace,  subject) | Replaces all occurences of ‘search’ with the ‘replace’ in the ‘subject’ string | strRep(“a”,”z”,”abca”)   * “zbcz” |

Notes

* Custom function names are case sensitive, whereas in-built functions are not.
* Certain things may be suffixed with ‘\_’ to prevent name collisions.
* Equations are usually assumed to only contain x and y as variables.
* Programs names are limited to 8 characters which hopefully explains bizarre naming choices.

License

The MIT License (MIT)

Copyright (c) 2015 Elliot Levin (elliotlevin@hotmail.com)

Permission is hereby granted, free of charge, to any person obtaining a copy  
of this software and associated documentation files (the "Software"), to deal  
in the Software without restriction, including without limitation the rights  
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
copies of the Software, and to permit persons to whom the Software is  
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in  
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR  
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,  
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE  
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER  
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,  
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN  
THE SOFTWARE.