Prof. R. Sorbello
Dept of Physics, UWM
April 2007

LIST OF MATHEMATICA COMMANDS AND EXAMPLES

FUNCTIONS

Examples of built-in Mathematica functions: Sin[x] , Cos[x], Tan[x], Log[x], Sinh[x], Cosh[x], ArcSin[x], ArcCos[x], ArcTan[x], Abs[x], Sign[x], LegendreP[n, x], BesselJ[m,kr], SphericalHarmonicY[l, m, theta, phi].

Example of defining your own function: F[x_]:= E^x Sin[a x] . The "F[x_] :=" construction allows you to call F repeatedly with different x values, just as you do for the built-in functions. Alternatively, you can write f = E^x Sin[a x] , in which case f gets an immediate assignment based on the value of x at the time you execute the command, and that specific value of x will remain in f unless you later clear x using the command Clear[x] or choose another value of x before executing f. An alternate construction is to define f= E^x Sin[a x] and when you need it for x = b, use z= f /. x->b

SPECIAL CONSTANTS

Pi, E, I, Infinity. (Note: Mathematica protects these names; they cannot be used as names for other quantities.)

DERIVATIVES

Let f be some expression containing x. (For example: f = x y + y^2 +Sin[b x])
First partial derivative of f with respect to x: D[f, x]
n-th partial derivative of f with respect to x: D[f, {x,n}]
Mixed partial-derivative: D[f, x, y] or D[D[f, x], y]

INTEGRALS

Indefinite integral: Integrate[f,x] (Here f is an expression depending on x)
Definite integral from x=x1 to x=x2: Integrate[f,{x,x1,x2}] (Here x1 and x2 may be symbolic variables)
Indefinite double Integral: Integrate[f, x,y] (Here f is an expression depending on x and y)
Definite double Integral: Integrate[f, {x,x1,x2},{y,y1,y2}]
Assumptions on parameters in an integral (See also ASSUMPTIONS section below):
If we know that a parameter, n, is in a certain domain, say n > 1, add-on the option: "Assumptions -> n > 1". Example: Integrate[1/x^n, {x,1,Infinity}, Assumptions -> n>1]
Example for multiple assumptions: Integrate[1/x^(n+m), {x,1,Infinity}, Assumptions -> {n>1, m>0}]
Numerical Integration (necessary if no analytic solution exists): NIntegrate[f,{x,x1,x2}], where now x1, x2 must be numerical quantities, and similarly for double integrals: NIntegrate[f, {x,x1,x2},{y,y1,y2}]

SUMS

Sum[f,{n,nmin,nmax}] , where for example f = 1/n^2 and nmin = 1 and nmax = Infinity.
Sum [f,{n,nmin,nmax,nstep}] , where n goes from "nmin" to "nmax" in steps of "nstep".
Double sum: Sum[f,{m,mmin,mmax},{n,nmin,nmax}]

SERIES EXPANSIONS

Series[f, {x, x0, n}] gives a Taylor series expansion for f as a function of x about the point x = x0 to order (x - x0)^n.
Series[f, {x, x0, n}, {y, y0, n}] gives a Taylor series expansion for f as a function of x and y about the point (x0,y0) to order n.
Series[f, {x, x0, n}, Assumptions->{a>0,b<0}] gives a Taylor series expansion for f subject to assumptions on the parameters a,b that appear in f. (See also ASSUMPTIONS section below).

PLOTS

Plot[f,{x,x1,x2}] plots specified f from x=x1 to x=x2.

Plot[F[x], {x,x1,x2}] plots user-defined function F[x] from x=x1 to x=x2.
Plot[{f,g,...},{x,x1,x2}] plots f, g,... from x=x1 to x=x2.
Plot3D[f, {x, xmin, xmax}, {y, ymin, ymax}] gives a 3-d plot of f(x,y).
ContourPlot[f, {x, xmin, xmax}, {y, ymin, ymax}, ContourShading -> False] gives contour plot of f(x,y).
DensityPlot is similar to ContourPlot but uses shading without contour lines.
ImplicitPlot for curve of f(x,y)=0 in xy-plane:
  << Graphics`ImplicitPlot`
  ImplicitPlot[f == 0, {x,x1,x2}, {y,y1,y2}]
Parametric plot of y(x) from defined x[t] and y[t]: ParametricPlot[{x[t], y[t]}, {t, 0, tmax}, AspectRatio -> Automatic]
To SHOW PLOT in different domains:
  myplot=Plot[Sin[x],{x,0,10}]
  Show[myplot,PlotRange->{{x1,x2},{y1,y2}}] (For all pts: PlotRange->All)
To SHOW several plots superimposed:
  Show[{myplot1,myplot2,myplot3},PlotRange->{{x1,x2},{y1,y2}}]
Multicolor plot of a family of functions: Plot[Evaluate[Table[Cos[n Pi x], {n,1,5}]], {x, -1, 1}, PlotStyle -> {Red, Yellow, Green, Blue, Orange}]
Useful Aliases (in what follows, f,f1,f2 slots are specified as functions of x:
  myplot[f_,x_,x1_,x2_]:= Plot[f,{x,x1,x2},PlotRange->All, AxesLabel->{x,f}]
  myplot2a[f1_,f2_,x1_,x2_]:=Plot[{f1, f2}, {x, x1, x2},PlotRange->All,PlotStyle->{Blue, Red}]
  myplot2b[f1_,f2_,x1_,x2_]:=Plot[{f1, f2}, {x, x1, x2},PlotRange->All,PlotStyle -> {Thickness[.003],Dashing[{.02,.015}]}]
  myplot3[f1_,f2_,f3_,x1_,x2_]:=Plot[{f1,f2,f3}, {x,x1,x2},PlotRange->All,PlotStyle->{Blue,Red, Green}]

LISTPLOTS
ListPlot[xflist,PlotJoined->True,PlotRange->All], where, xflist is a paired list of {x,f} entries (see "LISTS" below).
Useful Alias: mylistplot[data_]:=ListPlot[data,PlotJoined->True,PlotRange->All,AxesLabel->{"x","f"}]
  where, e.g., data=Table[{xval[i],fval[i]},{i,1,100,1}];
Multiple ListPlots shown together:
 << Graphics`MultipleListPlot`;
 MultipleListPlot[list1, list2, PlotJoined->{True, False},PlotRange->All]

PLOT OPTIONS
Plot[f,{x,x1,x2}, option1, option2, ...], where some options follow:
To restrict plotted values between f = c1 and f = c2: PlotRange -> {c1,c2}
To plot all of values of f in the interval (if Mathematica fails to): PlotRange -> All
To increase the number of sampled plot points used to 50: PlotPoints -> 50
For labels on x-axis and y-axis: AxesLabel -> {"x", "f"}
For dashed line: PlotStyle -> Dashing[{0.02,0.015}]
For 2 curves (full and dashed): PlotStyle -> {Thickness[.003],Dashing[{.02,.015}]}
For colored line: PlotStyle -> Blue or Red, Green, Yellow, Orange, Purple, or Black (default).
For 2 curves of different color: PlotStyle -> {Blue,Red}
For 2 curves of different color, thickness, dashing (can omit any options): PlotStyle -> {{Thickness[0.008], Blue}, {Thickness[0.01], Dashing[{0.012, 0.015}], Red}}
For plot label: PlotLabel -> FontForm["My Graph", {"Helvetica-Bold", 12}]
For grid-lines on plot: GridLines -> Automatic
For border around plot: Frame -> True
For no ticks on frame: FrameTicks -> None
To set font for all future graphics text: $TextStyle = {FontFamily -> "Times", FontSize -> 12}
To change Plot3D viewpoint to coordinates x1,x2,x3: ViewPoint-> {x1,x2,x3}
To eliminate shading in the ContourPlot: ContourShading -> False

FIELD PLOTS
To create a vector-field plot of grad f(x,y) from x0 to x1 and y0 to y1, load the graphics plot package by typing: Needs["Graphics`PlotField`"]

Then type: PlotGradientField[f, {x, x0, x1}, {y, y0, y1}]
or PlotGradientField[f, {x, x0, x1}, {y, y0, y1}, HeadWidth->0.01, HeadLength->0.01, HeadCenter->0]
(Caution: This is a plot of Grad[f], but it is not a conventional field plot. The magnitude of the field is indicated by lengths of arrows rather than the density of field lines.)

SIMPLIFY
To simplify algebraic expression for f: Simplify[f]
To really simplify, try: FullSimplify[f].
To simplify, making assumptions on domain of variables in expression, see ASSUMPTIONS section below.

ASSUMPTIONS
To make assumptions on variables in expressions used in commands Simplify, FullSimplify, Integrate:

Simplify[expr, Assumptions-> ...], where ... = a>0, or 1>a>0, or Im[a]==0, or Element[p, Integers], or Element[x, Reals], or Element[x,y,x, Reals], etc. For multiple assumptions, either enclose by braces: Simplify[expr, Assumptions->{...}], where {...} = {a>0, b>0, c>0}, etc., or use logical-and construction without braces: Simplify[expr, Assumptions->...], where ... = a>0&&b>0&&c>0, etc.

For Mathematica 5.2 and later: Series command with assumptions also works, as in Series[f,{x,0,3}, Assumptions-> a>0}], and in the special cases of Simplify, FullSimplify and Refine, can drop "Assumptions-> " and just write the assumption, as in Simplify[f, a>0] or Simplify[f, a>0&&b>0].

Refine[expr, assum] gives the form of expr that would be obtained if symbols in it were replaced by explicit numerical expressions satisfying the assumptions assum. Example: Refine[Log[x], x<0] gives I*Pi + Log[-x]

EVALUATING EXPRESSIONS
To evaluate an expression f numerically: N[f]
To evaluate f to an accuracy of M decimal-places: N[f,M]
To evaluate f where f depends on x and y, for the case of a particular value of y, use the replacement operator " /." as follows:
G = f /. y -> 2 makes G equal to f evaluated at y = 2, but leaves f as before. (If use y = 2 followed by G = f, then would have to use "Clear[y]" to reconstruct functional form of f.)
To perform several replacements: G = f /. {y -> 2, a -> 5, b -> z^2}
Alternatively, you could have defined f originally as f[x_, y_] := ..., in which case writing G = f[x,2] does not affect the original expression for f[x,y].

CLEAR
Clear assigned value for variable x: Clear[x] or type: x =.
Clear assigned values for variables x, a,... : Clear[x,a,...]
Clear def'n for function f and values x,a,... : Clear[f,x,a,...]

COMPLEX VARIABLES
ComplexExpand[expr] will give real and imaginary parts of expr assuming all variables (if any) are real
ComplexExpand[expr,{x1,x2,...}] expands expr assuming all variables are real except x1,x2,... which are complex
Useful commands: Re[z], Im[z], Abs[z], Conjugate[z], Arg[z]
Load package to help simplify complex expressions: Algebra`ReIm`
Declare the symbol z to be real: z /:Im[z]=0
Declare functions f,g, ... to be real-valued for real arguments: RealValued[f, g, ...]
Declare the symbol a to be negative: Negative[a] ^= True

DISPLAYING MATH OUTPUT
If output from a certain command "expr" is not wanted, type: expr ;

To show a one-line outline form of the output, type: expr //Short

VECTOR ANALYSIS
Load this package by typing: Needs["Calculus`VectorAnalysis`"]
Choose spherical coordinate system by typing: SetCoordinates[Spherical[r,theta,phi]]
Choose cartesian coordinate system by typing: SetCoordinates[Cartesian[x,y,z]]
Choose cylindrical coordinate system by typing: SetCoordinates[Cylindrical[r,phi,z]]
Vector operations: Grad[f], Div[f], Curl[f], CrossProduct[v1,v2], v1 . v2, where vector fields f, v1, v2 are specified as { , , }, and v1 . v2 gives the dot product.
Laplacian of a function g: Laplacian[g]

TABULATE A FUNCTION
To make a table of x and F[x] or a table of x and f from x1 to x2 in increments of xstep:
Table[{x, F[x]},{x, x1, x2, xstep}]//TableForm
Table[{x, f},{x, x1, x2, xstep}]//TableForm
Useful alias: tf[mylist_]:= mylist//TableForm , where mylist=Table[...] of above examples.

IMPORTING AND EXPORTING DATAFILES AND GRAPHICS
Files are placed in a default directory (usually user's home directory).
To find this directory: Directory[]
To change this directory: SetDirectory["pathname"], where, e.g., pathname= /users/home/work.

To Export datafiles (these datafiles must have the ".dat" suffix):
Create a datalist or Table, e.g., mylist=Table[{x,Sin[x],Cos[x]},{x, 0, 7, 0.1}]
To create a datafile (myfile.dat) from the above list:
Export["myfile.dat", mylist],
where myfile.dat consists of 71 lines each with values of x, Sin[x], Cos[x] separated by spaces.

To Import a datafile as a list:
zlist= Import["myfile.dat"], where myfile.dat contains lines of data with numbers on each line separated by space(s).
To create 3 1-d arrays from above list consisting of nmax=71 lines of data:
Do[xarray[i] = zlist[[i,1]]; sinarray[i]=zlist[[i,2]], cosarray[i]=zlist[[i,3]],{i, 1, nmax}]

To create a new file and put an expression (mathematica output) in the file wiping out previous entries in the file:
expr >>myfile (same as Put[data, "myfile"])
To append an expression to an existing file: expr >>>myfile (same as PutAppend[data, "myfile"])

To Export mathematica graphics into Word:
Mathematica: Edit/Save Selection As...EPS
Then in Word: Edit/Insert Picture...from file (and browse to the saved EPS file)

PRINTOUT
Given a Table or List: zlist=Table[{x,Sin[x],Cos[x]},{x,0,Pi,0.1}]
If want printout in table form: zlist//TableForm
If want printout via print statements as list is created:
Do[Print[x," ",Sin[x]," ",Cos[x]], {x,0,Pi,0.1}];
To prettify Print output to remove jags in column alignment:
PaddedForm[expr, {nt,nr}] gives nt total digit spaces with nr digits to right of dec. pt.:
pf[w_]:=PaddedForm[w, {10,6}];
Do[Print[pf[x],pf[Sin[x]],pf[Cos[x]]], {x,0,Pi,0.1}]

OPTIONS
To find current options for a given command, like Plot: Options[Plot]
To find a specific option, like PlotRange: Options[Plot, PlotRange]
Change default options for session: SetOptions[Plot, PlotRange->All]

Determining a specific option as implimented: AbsoluteOptions[myplot, PlotRange]
Some options for Plot: AxesOrigin-> {0,0}, GridLines-> Automatic

HELP
In addtion to Mathematica's "HELP" menu, can get help on a command, by typing ?Command or ??Command for greater detail. For example: ??Integrate.

MORE ADVANCED TOPICS

DO LOOP
Do[expr,{i, imin, imax}]
Do[{expr1, expr2, expr3}, {i,imin,imax,di}]
Do[{expr1; expr2; expr3}, {i,imin,imax,di}]
Do[expr1; expr2; expr3, {i,imin,imax,di}]
Do[{expr1; expr2; expr3}, {i,imin,imax},{j,jmin,jmax}]
Caution: In above double iteration, the sum over j is done first for each fixed i, then i is summed-over last.

WHILE LOOP
While[(n<nmax), {expr1, expr2, expr3}]
While[(n<nmax), {expr1; expr2; expr3}]
While[(n<nmax), expr1; expr2; expr3]

CONDITIONALS
If[z>a,x=x1] (If z>a then x=x1, otherwise do nothing.)
If[z>a, x=x1,x=x2] (If z>a then x=x1, else x=x2)
If[z == 0 , x=x1,x=x2] (If z=0 then x=x1, else x=x2)
If[z>a && z<b , x=x1,x=x2] (If z>a AND z<b, x=x1, else x=x2)
If[z>a || z<b , x=x1,x=x2] (If z>a OR z<b, x=x1, else x=x2)
Which[test1,value1,test2,value2...] (Gives value corresp. to first true test)

ANIMATION EXAMPLES
Needs["Graphics`Animation`"]; (* To load the animation package *)
Animate[Plot3D[Sin[2 x] Sin[3 y] Cos [t],{x,0,Pi},{y,0,Pi},Axes -> False, PlotRange -> {-1.0,1.0}, DisplayFunction -> Identity], {t, 0, 2 Pi}]
(To animate a sequence of plots created by Animate command, double-click on the first plot in the cell.)
Create a sequence of plots in a cell by a Do-Loop (as in this trajectory example):
x[t_] := -Cos[t] + t; y[t_] := Sin[t];
Do[ParametricPlot[{x[t], y[t]}, {t, 0, trun}, AspectRatio -> Automatic, Axes -> False, PlotRange -> {{0, 40}, {-2, 2}}], {trun, 0, 40, 0.5}]
Then select the block of plots and choose Cell/Animate Selected Graphics.

LISTS, MAKING & PLOTTING
data=Table[{x,Sin[x]},{x,0,2 Pi,.2}] (* This makes a list of paired entries *)
data//TableForm (* This displays list in a table form *)
A useful alias: tf[mylist_]:=mylist//TableForm (* Then can use tf[data] instead of data//TableForm*)
ListPlot[data] (* This puts data points on plot *)
ListPlot[data,PlotJoined->True] (* To connect data points on plot *)

xdata=Table[x, {x,0,1,.1}]
ydata=Table[Sin[x], {x,0,1,.1}]
alldata=Table[{xdata[[i]],ydata[[i]]},{i,1,11}]
alldata//TableForm
ListPlot[alldata]

LISTS AND ARRAYS

flist=Table[farray[i],{i,1,imax}] (*To make a 1d-list from a 1d-array*)
Do[farray[i]=flist[[i]],{i,1,imax}] (*To make a 1d-array from a 1d-list*)
Do[xarray[i]=xlist[[i]],{i,1,imax}] (*To make another 1d-array from a 1d-list*)
xflist=Table[{xarray[i],farray[i]},{i,1,imax}] (*To make a paired-list from two 1-d arrays*)
xflist=Table[{xlist[[i]],flist[[i]]},{i,1,imax}] (*To make a paired-list from two 1-d lists*)
ListPlot[xflist,PlotJoined->True] (*To plot above list of paired elements *)
Do[{xarray[i]=xflist[[i,1]];farray[i]=xfist[[i,2]],{i,1,imax}] (*To make 2 arrays from 2d-list*)

xlist=Table[x,{x,0,1,.1}] (*To make a 1d-list from a function*)
flist=Table[f[x],{x,0,1,.1}] (*To make a 1d-list from a function*)
mydata=Transpose[{xlist,flist}] (* To make a list of paired entries {{x1,f1},{x2,f2}, ...} *)
ListPlot[mydata,PlotJoined->True] (*To plot above list of paired elements *)

To make array from a function: Do[x=x1+(i-1)*(x2-x1)/(imax-1); farray[i]=f[x], {i,1,imax}]

Flatten: If each entry of a list "strangelist" is nested inside brackets {...}, to remove these internal brackets:
list=Flatten[strangelist]

Note: Each element of an array must be a number, so if it turns out
(from a module calc, say) that each value of farray[i] is a one-number list, and so having brackets around this number, then in above commands need to replace farray[i] by {farray[i]} . This undoes brackets of f[x]. (Related example: y={.2}; {yvalue}=y (* This gives yvalue=.2 *)

To plot an array A[i,j]:
ListPlot3D[Table[A[i,j],{i,imin,imax},{j,jmin,jmax}]]
Could also make a function from an array, and then plot it as a list:
m[h_,T_]:=marray[IntegerPart[(h-hbeg)/dh], IntegerPart[(T-Tbeg)/dT]];
myplot=ListPlot3D[Table[m[h,T],{h, -0.5,0.5,0.025},{T,0.001,2.0,0.02}], Ticks->None,AxesLabel->{T,H,M}]

List Extraction Rules:
If: sol= {{x -> a, y -> b}, {x -> c, y -> d}}
Then: a= sol[[1,1,2]], b= sol[[1,2,2]], c=sol[[2,1,2]], d=sol[[2,2,2]]
(* sol[[m,n,p]] gives m-th element, n-th sub-element, p-th sub-sub-element *)
Example: sol= {{x[t] -> Cos[t]}}
Then use: Plot[sol[[1,1,2]],{t,0,Pi}] (*This fishes-out Cos[t] from sol list*)
If: sol= {{x -> a}, {x -> b}, {x -> c}, {x -> d}}
Then: a=sol[[1,1,2]], b=sol[[2,1,2]], c=sol[[3,1,2]], d=sol[[4,1,2]]
Example (solving 4 simultaneous eqs in unknowns x,y,w,z and fishing-out x :):
sol=Solve[{LHS1==0,LHS2==0,LHS3==0,LHS4==0},{x,y,w,z}]; xresult = sol[[1,1,2]]

To Obtain an Interpolating Function from a List:
flist=Table[Sin[x],{x, 0, 2*Pi, 0.05}]; xlist=Table[x, {x, 0, 2*Pi, 0.05}];
f=ListInterpolation[flist,{xlist}];Plot[f[x],{x ,0 , Pi}]
(* Note the ftn call as f[x] although it was created from "f=ListInterpolation..." without x-variable written. The x-variable in f[x] is associated with {xlist}. *)

SYSTEMS OF EQUATIONS

Simultaneous eqs.:
sol=Solve[{eq1==0,eq2==0},{x,y}] (use NSolve for numerical sol'n)
expr /. sol (This evaluates expr using solution values)

Differential eqs:
DSolve[{eqn1, eqn2,...},y[x],x] (for analytical solutions)
NDSolve[{eqn1,eqn2,...},y,{x, xmin, xmax}] (for numerical solutions)

Using Numerical solutions:
mysol = y[x] /. NDSolve[{y''[x]+y[x]==0,y[0]==1.,y'[0]==0},y[x],{x,0,Pi}]
This command makes mysol the solution y[x], otherwise the ND command simply returns
{y-> Interpolating ftn}, which is not such a useful object.

MATRICES and MATRIX EQUATIONS
Enter Matrices and N-dimensional vectors from the input palette using the 2-by-2 small-box array surrounded by parentheses for a matrix, and the vertical array of 2 small boxes surrounded by parentheses for a vector. To add additional rows, press the return key while holding the control key down (control-return). To add additional columns press the comma key while holding the control key down (control-,).

For matrix multiplication use a period. For example, M1.M2 where M1 and M2 are matrices.
Useful commands: Inverse[M], Det[M], Tr[M], Transpose[M], ConjugateTranspose[M], MatrixPower[M, n].
To display matrix M as columns and rows, use M//MatrixForm or M//TraditionalForm. Otherwise, Mathematica will display matrix as a list of lists.
To display vector X as a column, use X//MatrixForm or X//TraditionalForm.
To display matrices and vectors as columns and rows throughout session, choose Cell/Default Output Type/TraditionalForm.

Using matrices to solve a set of linear, inhomogeneous eqs: Express eqs as Y=M.X, where Y is a known N-dimensional vector and M is a known N-by-N matrix. Solve for unknown N-dimensional vector X, using the command X=Inverse[M].Y

PROGRAMMING
A simple integration program:
```
F[x_]:=x^2
mysum=0.0; x1=0.0; x2=10.0; npts=1001;
dx= (x2-x1)/(npts-1);
Do[
x=x1 + (n-1)*dx;
mysum=mysum + F[x],  {n,1,npts}];
integralval = mysum*dx;
Print["Integral= ", integralval]
```

Modules (sub-routines)
Module[{x, y, ...}, expr1; expr2; exprlast] specifies that occurrences of the symbols x, y, ...in expr should be treated as local. Evaluation of exprlast is value taken for module ftn. The following is a "module" (subroutine) of the previous program. One can specify x1, x2, and npts in a sub-routine call (assuming F[x] already has been defined). To integrate a pre-existing F[x] function using this module, just type, for example: myintegrate[0,10,1001], where we defined:

```
myintegrate[x1_,x2_,npts_]:=Module[{n,mysum,x,dx,integralval},
mysum=0.0;
dx= (x2-x1)/(npts-1);
Do[{
x=x1 + (n-1)*dx,
mysum = mysum + F[x]},
{n,1,npts}];
integralval=mysum*dx
];
```

Defining a piecewise-continuous function using step-function windows:
```
  step[x_]:=(1+Sign[x])/2;
  window[x_,x1_,x2_]:=step[x-x1] step[x2-x];
  F[x_]:=f1*window[x,0,a]+f2*window[x,a,b]
```
Defining a piecewise-continuous function using "Which" (see "CONDITIONALS" above):

F[x_]:=Which[x<0, 0, x>0 && x<a, f1, x>a && x<b, f2, x>b, 0]

MISCELLANY

KEYBOARD SHORTCUTS

Greek symbols: esc-g-esc for gamma, esc-G-esc for capital gamma, etc.

Symbol shortcuts based on TeX language (for Mathematica Notebooks): esc-\TeX-esc, where \TeX denotes the TeX command for a single Greek or mathematical symbol. For example, \TeX= \gamma, \Gamma, \hbar, \partial, \times, \sum, \int, \propto, \rightarrow, \sim, \equiv, \cdot, \perp, \parallel.

Exponents: x to the power y can be entered as x control-^ y.

TeX

Mathematica to TeX conversion: TeXForm[expr]
Tex to Mathematica conversion: ToExpression["input",TeXForm], where, for example, input = \sqrt{b^2-4ac}.

NOTEBOOK PAGE BREAKS

To see them: Format/Show PageBreaks

To return to notebook view: Format/Show PageBreaks (toggle switch)

To create or remove pagebreaks, use the page break palette. (To get it, go to www.wolfram.com. and in the Search Site box, type: page break palette.) Alternatively, go to Mathematica/Preferences, and choose Cell Options/PageBreaking (Be sure to use "Selection" in top bar of Option Inspector, unless want entire document to be reformated.)

FIXING COMMON MATHEMATICA ERRORS

Use curly brackets "{" and "}" to enclose a list or group of similar objects in commands, and use square brackets "[" and "]" to enclose arguments of functions or the entire set of objects or arguments appearing in a command.

When use function construction such as "F[x_,y_]:= y*Sin[x] " be sure to use underscore after each argument on left hand side only.

Avoid using non-desired values for functions and variables in subsequent expressions. Fewer problems are likely to occur in multiple uses of defined expressions if use delayed assignment with ":=" or if evaluate an expression with the substitution command as in "f/. x->a".

In a "DO loop" each command is followed by ";" except that the last command before the loop-index brackets is followed by ","

Be careful about missing "}" or ";" or "]" in Do-loops, missing "," and "]" in Print statements, and "}", "{" and ","in lists.

If Mathematica's instruction alignment in a block of commands looks bad, one of the above rules is probably being violated.

If Mathematica gives error message about Tags and Protected quantities, check for violations of above rules.

Mathematica error messages such as "f is not a machine-size real number at x=..." in response to a Plot command are often due to undefined variables, or missing "," or ";" as for example a set of commands inside a Do-loop in which a Print statement is immediately followed by a Plot command without a ";"

between them.

Comment: This webpage is an evolving list of Mathematica commands, examples, tricks and Mathematica frustration-avoidance measures that I have been accumulating over the years in my work as a physicist and teacher. This is intended to be a quick-reference help-file that one might consult instead of, or prior to, accessing Mathematica's extensive Help menu. I am posting this webpage so that students and others are able to use it at their desktops from any connection to the web. (This document can be located by Googling "sorbello mathematica".) For Mathematica beginners, I have also posted a bare-bones guide to Mathematica at http://www.uwm.edu/~sorbello/classes/mathematica_primer.pdf.

Prof. Richard S. Sorbello
Department of Physics
University of Wisconsin-Milwaukee
Milwaukee, WI 53211
Email: sorbello@uwm.edu

Most recent updates: Jan 06, Nov 06, Apr 07