

In[] and Out[]

In[] and Out[] offers readers an opportunity to ask questions of the experts. The *Journal* encourages readers to submit problems in care of the editor.

Edited by Paul Abbott

Collect using a Parameter

Q: After defining

```
b[i_, j_]:= 1 - 2 a[i, j]
```

and expanding the product

```
expr = b[1,2] b[1,3] b[1,4] b[1,5] *
      b[2,3] b[2,4] b[2,5] // Expand;
```

how can I collect the terms of expr with the same number of a[i,j] factors.

Jorma Virtamo (jorma.virtamo@vtt.fi) suggests introducing a parameter z in the definition of b:

```
In[1]:= b[i_, j_]:= 1 - 2 a[i, j] z
```

Then compute the product:

```
In[2]:= expr = b[1,2] b[1,3] b[1,4] b[1,5] *
          b[2,3] b[2,4] b[2,5] // Expand;
```

Terms of expr with the same number of a[i,j] factors can be projected out using Coefficient:

```
In[3]:= aterms[n_] := Coefficient[expr, z^n]
```

```
In[4]:= aterms[1]
```

```
Out[4]= -2 a[1, 2] - 2 a[1, 3] - 2 a[1, 4] - 2 a[1, 5] -
        2 a[2, 3] - 2 a[2, 4] - 2 a[2, 5]
```

Allan Hayes points out that CoefficientList can be used to get the complete list of z coefficients in one go:

```
In[5]:= coefs = CoefficientList[expr, z];
```

The coefficient of z^n is coefs[[n+1]]. For example, here is the part of expr with seven a factors:

```
In[6]:= coefs[[8]]
```

```
Out[6]= -128 a[1, 2] a[1, 3] a[1, 4] a[1, 5] a[2, 3] a[2, 4]
        a[2, 5]
```

Integral Equation to ODE

Q: How can I solve the following functional equation?

$$f(x) = x + x \int_x^a (x^2 - pz) f(z) dz, \quad 0 < x < a$$

The integral equation

```
In[1]:= eqn = f[x] == x + x Integrate[(z^2 - p z) f[z], {z, a, x}];
```

can be turned into a first order linear ordinary differential equation by rearranging the terms so that the right-hand side contains only the integral,

```
In[2]:= (# - x)/x & /@ eqn
```

```
Out[2]= 
$$\frac{-x + f[x]}{x} == \text{Integrate}[(-(p z) + z^2) f[z], \{z, a, x\}]$$

```

and differentiating both sides of the equation:

```
In[3]:= D[%, x] // Simplify
```

```
Out[3]= 
$$\frac{-f[x] + x f'[x]}{x^2} == x (-p + x) f[x]$$

```

Noting the boundary condition f[a] == a, one obtains a differential equation that is immediately solvable:

```
In[4]:= soln = DSolve[{f[a] == a}, f[x], x] //
        First // ExpandAll
```

```
Out[4]= {f[x] -> E-a /4 + (a3 p)/3 - (p x3)/3 + x4 /4 x}
```

It is easy to check that the boundary condition is satisfied,

```
In[5]:= soln /. x -> a
```

```
Out[5]= {f[a] -> a}
```

and that the solution satisfies the original equation:

```
In[6]:= eqn /. f -> Function[x, Evaluate[f[x] /. soln]] // ExpandAll
```

```
Out[6]= True
```

Surface Area of a Doughnut

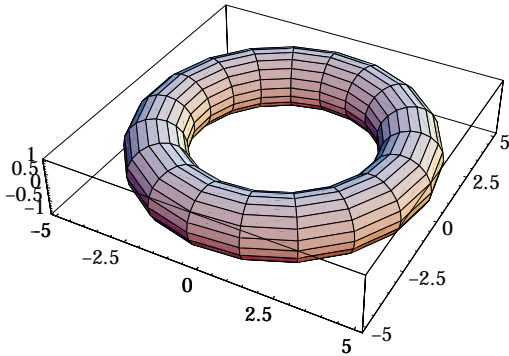
Q: I guessed that the surface area of a doughnut (torus) is given by $2\pi^2(r_o^2 - r_i^2)$, where r_o and r_i are the outer and inner radii, by noting that $R = (r_o + r_i)$ is the diameter of the dough and $r = (r_o - r_i)/2$ is the average of the radii, so the area should be given by $(2\pi r)(2\pi R)$. How can I derive the formula using calculus?

A general method for computing the surface area was outlined by Zdislav V. Kovarik (kovarik@mcmaster.ca). First, express the surface parametrically,

```
In[1]:= x[u_, v_, R_, r_] := (R + r Cos[u]) Cos[v]
In[2]:= y[u_, v_, R_, r_] := (R + r Cos[u]) Sin[v]
In[3]:= z[u_, v_, R_, r_] := r Sin[u]
```

where the parameters u and v range from 0 to 2π . The torus can be visualized using `ParametricPlot3D`:

```
In[4]:= Torus[R_, r_] := ParametricPlot3D[
  {x[u,v,R,r], y[u,v,R,r], z[u,v,R,r]},
  {u, 0, 2Pi}, {v, 0, 2Pi}]
In[5]:= Torus[4, 1]
```



Next, calculate the vector partial derivatives:

$$\mathbf{T}_u = \frac{\partial}{\partial u} \{x, y, z\}, \quad \mathbf{T}_v = \frac{\partial}{\partial v} \{x, y, z\}$$

In *Mathematica* this can be implemented as

```
In[6]:= Tu = D[#, u]& /@ {x[u,v,R,r], y[u,v,R,r], z[u,v,R,r]}
Out[6]:= {-(r Cos[v] Sin[u]), -(r Sin[u] Sin[v]), r Cos[u]}
In[7]:= Tv = D[#, v]& /@ {x[u,v,R,r], y[u,v,R,r], z[u,v,R,r]}
Out[7]:= {-(R + r Cos[u]) Sin[v], (R + r Cos[u]) Cos[v], 0}
```

The area element $dA(u, v)$ can be calculated from the cross product:

$$dA(u, v) = |\mathbf{T}_u \times \mathbf{T}_v|$$

After loading the package

```
In[8]:= << Calculus`VectorAnalysis`
```

the cross product can be evaluated:

```
In[9]:= CrossProduct[Tu, Tv]
Out[9]:= {-(r R Cos[u] Cos[v]) - r^2 Cos[u]^2 Cos[v],
  -(r R Cos[u] Sin[v]) - r^2 Cos[u]^2 Sin[v],
  -(r R Cos[v]^2 Sin[u]) - r^2 Cos[u] Cos[v]^2 Sin[u] -
  r R Sin[u] Sin[v]^2 - r^2 Cos[u] Sin[u] Sin[v]^2}
```

Computing the magnitude

```
In[10]:= % . % // Simplify
Out[10]:= r^2 (R + r Cos[u])^2
```

yields the area elements

```
In[11]:= dA[u_, v_] = PowerExpand[Sqrt[%]]
Out[11]:= r (R + r Cos[u])
```

Finally, integrating $dA(u, v)$ yields the surface area:

```
In[12]:= Integrate[dA[u, v], {u, 0, 2Pi}, {v, 0, 2Pi}]
Out[12]:= 4 Pi^2 r R
```

This formula agrees with the intuition about the areas of the torus and cylinder: When we take a cylinder of dough and bend it to form a doughnut, the area we shrink on the part of the surface close to the axis equals what we stretch on the part of the surface away from the axis.

Sqrt and ComplexExpand

Q: `ComplexExpand` expands its argument assuming that all variables are real. Why does the input

```
In[1]:= ComplexExpand[ Re [ Exp[ -I x Sqrt[y] ] ] ]
Out[1]:= E^x (y)^{2 1/4} Sin[Arg[y]/2] Cos[x (y)^{2 1/4} Cos[Arg[y]/2]]
```

not return `Cos[x Sqrt[y]]`? Since y is assumed to be real, I would have expected that `Arg[y]` is zero.

Daniel Lichtblau (danl@wri.com) answers: `ComplexExpand` cannot ascertain that `Arg[y]` is 0 because it might be π . For example,

```
In[2]:= Arg[-5]
Out[2]:= Pi
```

You can force a positive argument for `Sqrt` by using `Abs`:

```
In[3]:= ComplexExpand[ Re [Exp[ -I x Sqrt[Abs[y]] ] ] ] //
  PowerExpand
Out[3]:= Cos[x Sqrt[y]]
```

Swap File

Q: By default, Mathematica for DOS creates its swap file in the root directory of the drive on which it is installed. Is there any way to force the swap file to be in another directory?

John Fultz (Wolfram Research Technical Support) answers: This is documented in the *User's Guide for MS-DOS Systems*, which comes with *Mathematica* (Versions 2.2 and later), on pages 39 and 42. At the DOS prompt, change into the *Mathematica* directory and type the following:

```
CFG386 MATHEXE.EXE -SWD D:
```

where *D*: is the drive (you can make it any drive except for a RAM drive).

To check that the option has been added correctly, you can view all currently applied options:

```
CFG386 MATHEXE.EXE
```

Point Size in ScatterPlot3D

Q: Is there a way to change the size of the dots used in ScatterPlot3D?

Ian Collier (ianc@wri.com) answers: Consider the following example. After loading

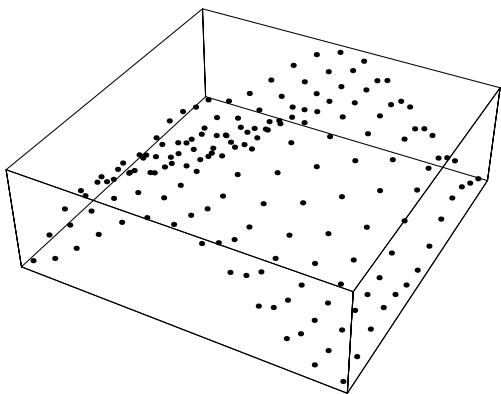
```
In[1]:= << Graphics`Graphics3D`
```

the following dataset

```
In[2]:= data = Flatten[Table[{x, y, Cos[x - Sin[y]] },  
  {x, -3, 3, 0.5}, {y, -3, 3, 0.5}], 1];
```

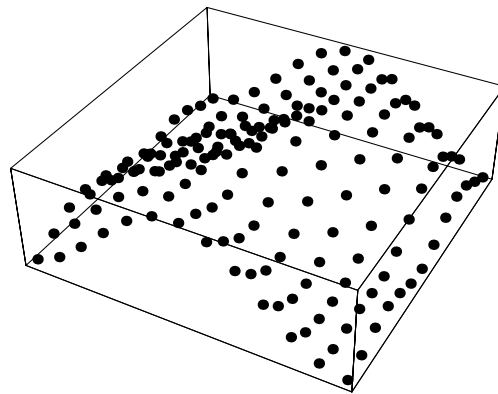
is plotted using ScatterPlot3D:

```
In[3]:= graph = ScatterPlot3D[ data ]
```



You can use `Insert` to place a `PointSize` graphics directive at the appropriate position in the `Graphics3D` object returned by `ScatterPlot3D`:

```
In[4]:= Show[Insert[graph, PointSize[0.02], {1, 1}]]
```



Batch

Q: I want to run Mathematica as a background process on a UNIX machine. However, the UNIX command

```
math -run "<< file.m"
```

does not work, even though the Mathematica input

```
<< file.m
```

works as expected when invoked in an interactive session.

To run a *Mathematica* job in the background, use the syntax:

```
math < file.m > file.out &
```

To get a full session log, including both the input and output, include the command `AppendTo[$Echo, "stdout"]`. For example, if the `file.m` contains

```
AppendTo[$Echo, "stdout"]  
Integrate[x Sin[x], x]  
D[%, x] // Simplify
```

then the resulting `file.out` is

```
Mathematica 2.2 for DEC OSF/1 Alpha  
Copyright 1988-94 Wolfram Research, Inc.  
-- Terminal graphics initialized --
```

```
In[1]:=  
Out[1]= {stdout}  
  
In[2]:= Integrate[x Sin[x], x]  
Out[2]= -(x Cos[x]) + Sin[x]  
  
In[3]:= D[%, x] // Simplify  
Out[3]= x Sin[x]  
  
In[4]:=
```

If you want just the output (in `InputForm` for re-usability), you can use

```
math -batchinput -batchoutput < file.m > file.out &
```

For this file.m,

```
Integrate[x Sin[x], x]
D[%, x] // Simplify
```

the resulting file.out is

```
-(x*Cos[x]) + Sin[x]
x*Sin[x]
```

Piecewise Differentiation

Q: *For a piecewise function,*

```
In[1]:= h[x_/;x>0] := Sin[x]
```

```
In[2]:= h[x_/;x<=0] := 0
```

how can I force the evaluation of the derivative of h?
Presently it is not evaluated:

```
In[3]:= h'[3]
```

```
Out[3]= h'[3]
```

Dave Wagner (dbwagner@princon.com) answers: There's a nice trick of using `Which` to define a piecewise function:

```
In[4]:= h[x_] := Which[x > 0, Sin[x], x <= 0, 0]
```

```
In[5]:= h'[3]
```

```
Out[5]= Cos[3]
```

```
In[6]:= h'[x]
```

```
Out[6]= Which[x > 0, 1 Cos[x], x <= 0, 0]
```

Unfortunately, `Integrate` doesn't understand such functions:

```
In[7]:= Integrate[h[x], x]
```

```
Out[7]= Integrate[Which[x > 0, Sin[x], x <= 0, 0], x]
```

You can `NIntegrate` such functions (apart from warnings about convergence):

```
In[8]:= NIntegrate[h[x], {x, -5, Pi/2}]
```

```
NIntegrate::ncvb:
```

```
NIntegrate failed to converge to prescribed accuracy
after 7 recursive bisections in x near x = 0.00509876.
```

```
Out[8]= 1.
```

Compile of Which

Q: I am trying to Compile a procedure that includes the compilable testing function Which:

```
In[1]:= f = Compile[{a, b},
  Which[ a < b, b,
        a > b, a,
        a == b, a ]];
```

However looking at the heart of the CompiledFunction (its third part):

```
In[2]:= InputForm[f][[1, 3]]
Out[2]= {{1, 17}, {4, 1, 0}, {4, 2, 1},
  {24, Function[{a, b},
    Which[a < b, b, a > b, a, a == b, a]], 2}, {8, 2}}
```

the opcode 24 indicates that Which has not been compiled and will be evaluated externally, which defeats the purpose of Compile and greatly slows down the calculation. Is it possible to Compile functions involving Which?

Kevin Martin Leuthold (leuthold@symcom.math.uiuc.edu) answers: Compile will work on Which if the last test is True. So, you can rewrite your function as follows:

```
In[3]:= f = Compile[{a, b},
  Which[ a < b, b,
        a > b, a,
        True, a ]];
```

We verify that the function is fully compiled:

```
In[4]:= InputForm[f][[1, 3]]
Out[4]= {{1, 17}, {4, 1, 0}, {4, 2, 1}, {75, 0, 1, 0}, {69, 0, 3},
  {17, 1, 3}, {70, 7}, {75, 1, 0, 1}, {69, 1, 3},
  {17, 0, 2}, {70, 2}, {17, 0, 2}, {17, 2, 3}, {8, 3}}
```

Partial Differential Equation

Q: How can I solve this first order partial differential equation

$$(\alpha x^2 + \beta x + \gamma)u_x(x, y) + u_y(x, y) + \lambda u(x, y) = 0$$

where α , β , γ , and λ are constants?

The package

```
In[1]:= << Calculus`PDSolve1`
```

extends the functionality of DSolve, enabling it to handle certain classes of first-order partial differential equations. Entering the equation:

```
In[2]:= eqn = (a x^2 + b x + c) Derivative[1, 0][u][x, y] +
  Derivative[0, 1][u][x, y] + d u[x, y] == 0
Out[2]= d u[x, y] + u(0,1)[x, y] +
  (c + b x + a x2) u(1,0)[x, y] == 0
```

the solution is immediate:

```
In[3]:= soln = DSolve[%, u[x, y], {x, y}] // First
Out[3]= {u[x, y] ->
  Power[E, (2 d ArcTanh[(b + 2 a x)/Sqrt[b2 - 4 a c]])/
  Sqrt[b2 - 4 a c]]
  2 ArcTanh[-----]
  Sqrt[b2 - 4 a c]
  C[1][y + -----]}
  Sqrt[b2 - 4 a c]
```

Here C[1] represents an arbitrary function.

The solution is easily verified by turning it into a pure function (using Function), back-substitution, and collecting terms on each side of the equation over a common denominator:

```
In[4]:= Together /@
  (eqn /. u ->
  Function[{x, y}, Evaluate[u[x, y] /. soln]])
Out[4]= True
```