The effect of incremental Euler rotations on an Euler orientation.

Dave Cromley, November 2011

Notation: To keep the expressions less intimidating:
An angle around the x axis is x instead of ax. δx is a small angle.
COS(x) is Cx; SIN(x) is Sx; TAN(x) is Tx.
Dydx is the partial derivative of y with respect to (a rotation by) δx.

We start with [m], the matrix from the Euler orientation (x, y, z).

```
[ +CyCz          +CxSz+SxSyCz  +SxSz-CxSyCz  ]
[ -CySz          +CxCz-SxSySz  +SxCz+CxSySz  ]
[ +Sy            -SxCy         +CxCy         ]
```

(See http://en.wikipedia.org/wiki/Euler_angles)

[δm] is the matrix of an incremental rotation by angle (e.g.) δx.

```
[  1   0   0  ]
[  0  Cδx Sδx ]
[  0 -Sδx Cδx ]
```

Since δx is small, we can use:

```
[  1   0   0  ]
[  0   1  δx  ]
[  0  -δx  1  ]
```

[m'] is the matrix of the resulting Euler orientation (x', y', z').
The objective is to find x', y', and z'.

We do the following for each of δx, δy, and δz
START process A; use δx e.g.:
  We get the difference matrix [dm]
    [dm] = [m']-[m] = [m][δm]-[m] = [m]([δm]-I)
  We get 9 equations from the 9 elements of [m] and [dm]:
    ∂mij∂x*Dxdx + ∂mij∂y*Dydx + ∂mij∂z*Dzdx = dmij
  With these 9 equations in 3 unknowns, we can get Dxdx, Dydx, and Dzdx.
  By choosing the best mij and dmij, the computations are not difficult.
END process A

Having all 9 partial derivitives, we have
x' = x + Dxdx*δx + Dxdy*δy + Dxdz*δz
y' = y + Dydx*δx + Dydy*δy + Dydz*δz
y' = z + Dzdx*δx + Dzdy*δy + Dzdz*δz

---- Apply δx rotation to get Dxdx, Dydx, Dzdx ----------

-- [δx]-I matrix

```
[  0  0  0 ]
[  0  0 δx ]
[  0 -δx 0 ]
```

-- [m]([δx]-I)

```
[ 0              -SxSz+CxSyCz  +CxSz+SxSyCz  ]
[ 0              -SxCz-CxSySz  +CxCz-SxSySz  ]
[ 0              -CxCy         -SxCy         ]
```

m31: +CyDydx = 0
     Dydx = 0
m11: -SyCzDydx -CySzDzdx = 0
     -CySzDzdx = 0
     Dzdx = 0
m33: -SxCyDxdx -CxSyDydx = -SxCy
     -SxCxCyDxdx = -SxCxCy
     Dxdx = 1

```
---- Apply δy rotation to get Dxdy, Dydy, Dzdy ----------------

-- [δy]-I matrix
[  0  0 -δy ]
[  0  0  0  ]
[ δy  0  0  ]
-- [m]([δy]-I)
[  +SxSz-CxSyCz  0             -CyCz         ]
[  +SxCz+CxSySz  0             +CySz         ]
[  +CxCy         0             -Sy           ]

m31: CyDydy = +CxCy
     Dydy = Cx
m11: -SyCzDydy -CySzDzdy = +SxSz -CxSyCz
     -CxSyCz -CySzDzdy = +SxSz -CxSyCz
     -CySzDzdy = +SxSz -CxSyCz +CxSyCz
     Dzdy = -Sx/Cy
m33: -SxCyDxdy -CxSyDydy = -Sy
     -SxCyDxdy -CxCxSy = -Sy
     -SxCyDxdy = -Sy +CxCxSy
     -SxCyDxdy = -SxSxSy
     Dxdy = SxTy

---- Apply δz rotation to get Dxdz, Dydz, Dzdz ----------------

-- [δz]-I matrix
[  0 δz  0  ]
[-δz  0  0  ]
[  0  0  0  ]
-- [m]([δz]-I)
[  -CxSz-SxSyCz  +CyCz         0             ]
[  -CxCz+SxSySz  -CySz         0             ]
[  +SxCy         +Sy           0             ]

m31: CyDydz = +SxCy
     Dydz = +Sx
m11: -SyCzDydz -CySzDzdz = -CxSz-SxSyCz
     -CySzDzdz = -CxSz-SxSyCz
     -CySzDzdz = -CxSz +SxSyCz +SxSyCz
     +CyDzdz = +Cx
     Dzdz = +Cx/Cy
m33: -SxCyDxdz -CxSyDydz = 0
     -SxCyDxdz -SxCxSy = 0
     +CyDxdz = -CxSy
     Dxdz = -CxTy
```

---- Combine the 9 results ----------

Dxdx = 1
Dydx = 0
Dzdx = 0
Dxdy = +SxTy
Dydy = +Cx
Dzdy = -Sx/Cy
Dxdz = -CxTy
Dydz = +Sx
Dzdz = +Cx/Cy

x' = x +δx 1 +δy SxTy -δz CxTy

y' = y +δx 0 +δy Cx +δz Sx

z' = z +δx 0 -δy Sx/Cy +δz Cx/Cy

I was unable to begin the computations for this until I stumbled across Jed Margolin's "Euler Angle Functions". His results didn't work for me, forcing me to do this paper.
http://www.jmargolin.com/uvmath/euler.doc

The desire to do this in the first place comes from the existance of the "passionate" programmers at thinBasic.com and the amazing product (it's free) and the active forum. The features of thinBasic seem innumerable. And the interests of the forum community are innumerable.
http://thinbasic.com

My thinBasic demo program using these results is at
http://dbarc.net/dcgimbalrock.exe
This paper resides at
http://dbarc.net/dceulerrot.pdf