Orbital Elements
version 1.7

The Add-In OE.xlam (or OE.xla) contains the quasi-periodic approximation formulae for Earth's orbital elements from Laskar et al (1993) and Laskar et al (2004), and data from his nominal solution La93(0,1) from −20M to +10M, given every 1000 Julian years from J2000, available at this Bureau des Longitudes ftp site link.

The following symbols are used:

e	eccentricity
ϖ	longitude of the perihelion from the fixed equinox of J2000, in radians
i	inclination from the fixed ecliptic of J2000, in radians
Ω	longitude of the ascending node from the fixed ecliptic and equinox of J2000, in radians
k	$e\cos\varpi$
h	$e\sin\varpi$
q	$\sin(i/2)\cos\Omega$
p	$\sin(i/2)\sin\Omega$
ψ	general precession in longitude from J2000, in radians
ε	obliquity of the ecliptic, in radians
t	dynamical time measured in Julian millennia (365250 days) from J2000

There is a short description of each function in Excels function manager (the *fx* button to the left of the formula bar) under the category "User Defined", or "Orbital Elements" in Excel 2007 and 2010.
The functions can be entered in single cells, or as arrays (Ctrl+Shift+Enter for an array).

The following function names ending with "D" are derived from Laskar's Data files. The data is stored in memory and interpolated with degree 7 Lagrange polynomials for k, h, q, p, ψ, ε

rkD(t)	function returns k, using Lagrange polynomial
ihD(t)	function returns h, using Lagrange polynomial
eccD(t)	function returns eccentricity e, where $e = \sqrt{k^2+h^2}$
wfxD(t)	returns longitude of the perihelion from the fixed equinox of J2000 ϖ, where $\tan\varpi = h/k$
ecc_wfxD(t_1:t_j)	2× matrix array function returns eccentricity in the 1st cell and longitude of the perihelion in the 2nd. It can go in either direction (across or down), and needs to be at least a 2×2 matrix
rqD(t)	function returns q, using Lagrange polynomial
ipD(t)	function returns p, using Lagrange polynomial
incD(t)	function returns inclination i, where $\sin(i/2) = \sqrt{q^2+p^2}$
nodD(t)	function returns longitude of the node Ω, where $\tan\Omega = p/q$
inc_nodD(t_1:t_j)	2× matrix array function returns inclination in the 1st cell and longitude of the node in the 2nd. It can go in either direction (across or down), and needs to be at least a 2×2 matrix
phiD(t)	function returns general precession in longitude ψ, using Lagrange polynomial
epsD(t)	function returns obliquity ε, using Lagrange polynomial
wmvD(t)	function returns longitude of the perihelion from the moving equinox, $\varpi + \psi$

All of the add-in functions beginning with "D" are derivatives. The following derivative functions also ending with "D" are the mathematical derivatives of the degree 7 Lagrange polynomial interpolation of the data files.

DrkD(t) function returns the derivative of k, $\dot{k} = \frac{dk}{dt}$ using Lagrange polynomial

DihD(t) function returns the derivative of h, $\dot{h} = \frac{dh}{dt}$ using Lagrange polynomial

DeccD(t) function returns the derivative of eccentricity, $\frac{de}{dt} = \frac{k\dot{k} + h\dot{h}}{\sqrt{k^2 + h^2}}$

DwfxD(t) function returns the derivative of the perihelion, $\frac{d\varpi}{dt} = \frac{k\dot{h} - h\dot{k}}{k^2 + h^2}$

DrqD(t) function returns the derivative of q, $\dot{q} = \frac{dq}{dt}$ using Lagrange polynomial

DipD(t) function returns the derivative of p, $\dot{p} = \frac{dp}{dt}$ using Lagrange polynomial

DincD(t) function returns the derivative of inclination, $\frac{di}{dt} = 2\frac{q\dot{q} + p\dot{p}}{\sqrt{(1 - q^2 - p^2)(q^2 + p^2)}}$

DnodD(t) function returns the derivative of the node, $\frac{d\Omega}{dt} = \frac{q\dot{p} - p\dot{q}}{q^2 + p^2}$

DphiD(t) function returns the derivative of precession, $\frac{d\psi}{dt}$ using Lagrange polynomial

DepsD(t) function returns the derivative of obliquity, $\frac{d\varepsilon}{dt}$ using Lagrange polynomial

These following functions beginning with "I" retreive the raw data (9 files stored in memory) at Integer values of t. If t is passed to an "I" function as a non-integer, it will return the data at Int(t), the integer less than or equal to t. To display all 16 digits of the data files in Excel, download the add-in "Xnumbers v.6.0" from thetropicalevents.com and use the function dCStr (x, 16).

IrkD(t) function returns the data of k

IihD(t) function returns the data of h

IrqD(t) function returns the data of q

IipD(t) function returns the data of p

IphiD(t) function returns the data of precession, ψ

IepsD(t) function returns the data of obliquity, ε

IeccD(t) function returns the data of eccentricity, e

IwfxD(t) function returns the data of longitude of the perihelion from the fixed equinox of J2000, ϖ

IwmvD(t) function returns the data of longitude of the perihelion from the moving equinox, $\varpi + \psi$

These following functions ending with "A" are from Laskar's Approximation formulae, calculated from the 3× matrices of terms (a_k , b_k , c_k , in radians) stored in memory from the add-in.

rkA(t) function returns k, where $k = \sum_{k=1}^{26} b_k \cos(a_k t + c_k)$

ihA(t) function returns h, where $h = \sum_{k=1}^{26} b_k \sin(a_k t + c_k)$

eccA(t) function returns eccentricity e, where $e = \sqrt{k^2 + h^2}$

wfxA(t) function returns longitude of the perihelion ϖ, where $\tan \varpi = h/k$

ecc_wfxA(t_1:t_j) 2× matrix array function returns eccentricity in the 1st cell and longitude of the perihelion in the 2nd. It can go in either direction (across or down), and needs to be at least a 2×2 matrix

rqA(t) function returns q, where $q = \sum_{k=1}^{24} b_k \cos(a_k t + c_k)$

ipA(t) function returns p, where $p = \sum_{k=1}^{24} b_k \sin(a_k t + c_k)$

incA(t) function returns inclination i, where $\sin(i/2) = \sqrt{q^2 + p^2}$

nodA(t) function returns longitude of the node Ω, where $\tan \Omega = p/q$

inc_nodA(t_1:t_j) 2× matrix array function returns inclination in the 1st cell and longitude of the node in the 2nd. It can go in either direction (across or down), and needs to be at least a 2×2 matrix

raA(t) function returns the real coefficient A of the complex number $A + iB$ from the approximation formula for precession and obliquity, where $A = \sum_{k=1}^{34} b_k \cos(a_k t + c_k)$

ibA(t) function returns the imaginary coefficient B of the complex number $A + iB$ from the approximation formula for precession and obliquity, where $B = \sum_{k=1}^{34} b_k \sin(a_k t + c_k)$

phiA(t) function returns general precession in longitude ψ, where $\tan \psi = B/A$

epsA(t) function returns obliquity ε, where $\sin \varepsilon = \sqrt{A^2 + B^2}$

phi_epsA(t_1:t_j) 2× matrix array function returns precession in the 1st cell and obliquity in the 2nd. It can go in either direction (across or down), and needs to be at least a 2×2 matrix

wmvA(t) function returns longitude of the perihelion from the moving equinox, $\varpi + \psi$

DrkA(t) function returns the derivative of k, $\dot{k} = \frac{dk}{dt} = \sum_{k=1}^{26} -a_k b_k \sin(a_k t + c_k)$

DihA(t) function returns the derivative of h, $\dot{h} = \frac{dh}{dt} = \sum_{k=1}^{26} a_k b_k \cos(a_k t + c_k)$

Dk_hA(t_1:t_j) 2× matrix array function returns the derivative of k in the 1st cell and the derivative of h in the 2nd. It can go in either direction (across or down), and needs to be at least a 2×2 matrix

DeccA(t) function returns the derivative of eccentricity, $\frac{de}{dt} = \frac{k\dot{k} + h\dot{h}}{\sqrt{k^2 + h^2}}$

DwfxA(t) function returns the derivative of the perihelion, $\frac{d\varpi}{dt} = \frac{k\dot{h} - h\dot{k}}{k^2 + h^2}$

DrqA(t) function returns the derivative of q, $\dot{q} = \frac{dq}{dt} = \sum_{k=1}^{24} -a_k b_k \sin(a_k t + c_k)$

DipA(t) function returns the derivative of p, $\dot{p} = \frac{dp}{dt} = \sum_{k=1}^{24} a_k b_k \cos(a_k t + c_k)$

Dq_pA(t_1:t_j) 2× matrix array function returns the derivative of q in the 1st cell and the derivative of p in the 2nd. It can go in either direction (across or down), and needs to be at least a 2×2 matrix

DincA(t) function returns the derivative of inclination, $\frac{di}{dt} = 2\frac{q\dot{q} + p\dot{p}}{\sqrt{(1 - q^2 - p^2)(q^2 + p^2)}}$

DnodA(t) function returns the derivative of the node, $\frac{d\Omega}{dt} = \frac{q\dot{p} - p\dot{q}}{q^2 + p^2}$

rglA(t) function returns $\sqrt{\left(\frac{dq}{dt}\right)^2 + \left(\frac{dp}{dt}\right)^2}$

DraA(t) returns the derivative of the real coefficient A of the complex number A + iB from the approximation formula for precession and obliquity, $\dot{A} = \frac{dA}{dt} = \sum_{k=1}^{34} -a_k b_k \sin(a_k t + c_k)$

DibA(t) returns the derivative of the imaginary coefficient B of the complex number A + iB from the approximation formula for precession and obliquity, $\dot{B} = \frac{dB}{dt} = \sum_{k=1}^{34} a_k b_k \cos(a_k t + c_k)$

DphiA(t) function returns the derivative of precession, $\frac{d\psi}{dt} = \frac{A\dot{B} - B\dot{A}}{A^2 + B^2}$

DepsA(t) function returns the derivative of obliquity, $\frac{d\varepsilon}{dt} = \frac{A\dot{A} + B\dot{B}}{\sqrt{A^2 + B^2}}$

These following functions ending with "M" are also from Laskar's approximation formulae, but from a 3× Matrix of terms (a_k, b_k, c_k, in radians) that you store on your spreadsheet. Matrices can go in either direction (across or down).

rkM(t, 3× matrix) function returns k, where $k = \sum_{k=1}^{26} b_k \cos(a_k t + c_k)$

ihM(t, 3× matrix) function returns h, where $h = \sum_{k=1}^{26} b_k \sin(a_k t + c_k)$

eccM(t, 3× matrix) function returns eccentricity e, where $e = \sqrt{k^2 + h^2}$

wfxM(t, 3× matrix) function returns longitude of the perihelion ϖ, where $\tan \varpi = h/k$

rqM(t, 3× matrix) function returns q, where $q = \sum_{k=1}^{24} b_k \cos(a_k t + c_k)$

ipM(t, 3× matrix) function returns p, where $p = \sum_{k=1}^{24} b_k \sin(a_k t + c_k)$

incM(t, 3× matrix) function returns inclination i, where $\sin(i/2) = \sqrt{q^2 + p^2}$

nodM(t, 3× matrix) function returns longitude of the node Ω, where $\tan \Omega = p/q$

raM(t, 3× matrix) function returns the real coefficient A of the complex number $A + \mathrm{i}B$ from the approximation formula for precession and obliquity, where $A = \sum_{k=1}^{34} b_k \cos(a_k t + c_k)$

ibM(t, 3× matrix) function returns the imaginary coefficient B of the complex number $A + \mathrm{i}B$ from the approximation formula for precession and obliquity, where $B = \sum_{k=1}^{34} b_k \sin(a_k t + c_k)$

phiM(t, 3× matrix) function returns general precession in longitude ψ, where $\tan \psi = B/A$

epsM(t, 3× matrix) function returns obliquity ε, where $\sin \varepsilon = \sqrt{A^2 + B^2}$

wmvM(t, 3× matrix for ϖ, 3× matrix for ψ) function returns longitude of the perihelion from the moving equinox, $\varpi + \psi$

DrkM(t, 3× matrix) function returns the derivative of k, $\dot{k} = \frac{\mathrm{d}k}{\mathrm{d}t} = \sum_{k=1}^{26} -a_k b_k \sin(a_k t + c_k)$

DihM(t, 3× matrix) function returns the derivative of h, $\dot{h} = \frac{\mathrm{d}h}{\mathrm{d}t} = \sum_{k=1}^{26} a_k b_k \cos(a_k t + c_k)$

DeccM(t, 3× matrix) function returns the derivative of eccentricity, $\frac{\mathrm{d}e}{\mathrm{d}t} = \frac{k\dot{k} + h\dot{h}}{\sqrt{k^2 + h^2}}$

DwfxM(t, 3× matrix) function returns the derivative of the perihelion, $\frac{\mathrm{d}\varpi}{\mathrm{d}t} = \frac{k\dot{h} - h\dot{k}}{k^2 + h^2}$

DrqM(t, 3× matrix) function returns the derivative of q, $\dot{q} = \frac{\mathrm{d}q}{\mathrm{d}t} = \sum_{k=1}^{24} -a_k b_k \sin(a_k t + c_k)$

DipM(t, 3× matrix) function returns the derivative of p, $\dot{p} = \frac{\mathrm{d}p}{\mathrm{d}t} = \sum_{k=1}^{24} a_k b_k \cos(a_k t + c_k)$

DincM(t, 3× matrix) function returns the derivative of inclination, $\frac{\mathrm{d}i}{\mathrm{d}t} = 2\frac{q\dot{q} + p\dot{p}}{\sqrt{(1 - q^2 - p^2)(q^2 + p^2)}}$

DnodM(t, 3× matrix) function returns the derivative of the node, $\frac{\mathrm{d}\Omega}{\mathrm{d}t} = \frac{q\dot{p} - p\dot{q}}{q^2 + p^2}$

rglM(t, 3× matrix) function returns $\sqrt{\left(\frac{\mathrm{d}q}{\mathrm{d}t}\right)^2 + \left(\frac{\mathrm{d}p}{\mathrm{d}t}\right)^2}$

DraM(t, 3× matrix) returns the derivative of the real coefficient A of the complex number $A + \mathrm{i}B$ from the approximation formula for precession and obliquity, $\dot{A} = \frac{\mathrm{d}A}{\mathrm{d}t} = \sum_{k=1}^{34} -a_k b_k \sin(a_k t + c_k)$

DibM(t, 3× matrix) returns the derivative of the imaginary coefficient B of the complex number $A + \mathrm{i}B$ from the approximation formula for precession and obliquity, $\dot{B} = \frac{\mathrm{d}B}{\mathrm{d}t} = \sum_{k=1}^{34} a_k b_k \cos(a_k t + c_k)$

DphiM(t, 3× matrix) function returns the derivative of precession, $\frac{\mathrm{d}\psi}{\mathrm{d}t} = \frac{A\dot{B} - B\dot{A}}{A^2 + B^2}$

DepsM(t, 3× matrix) function returns the derivative of obliquity, $\frac{\mathrm{d}\varepsilon}{\mathrm{d}t} = \frac{A\dot{A} + B\dot{B}}{\sqrt{A^2 + B^2}}$

Approximation Formulae

precession ψ and obliquity ε from the nominal solution La93(0,1) in Laskar et al (1993) are given as the complex number $\chi = \sin\varepsilon\, e^{i p_A}$, where e = 2.718… (the base of the natural logarithm), i = $\sqrt{-1}$, and p_A is the precession. The quasi-periodic approximation for χ is given as $\chi(t) \approx \sum_{k=1}^{34} B_k\, e^{i(\nu_k t + \psi_k)}$, where the 34 terms ν_k, B_k, ψ_k are from table 2, ν_k given in arcseconds per Julian year, B_k is dimensionless, and ψ_k in degrees. The terms in this add-ins descriptions are represented as the 3 × 34 matrix a_k, b_k, c_k, where a_k is ν_k converted to radians per Julian millennia, b_k is B_k, and c_k is ψ_k converted to radians. χ is represented here as $A + iB$, so that the add-ins function descriptions relate to

$$A + iB = \sum_{k=1}^{34} b_k\, e^{i(a_k t + c_k)}, \text{ and } A + iB = \sin\varepsilon\, e^{i\psi}$$

eccentricity e and longitude of perihelion ϖ are given in Laskar et al (2004) as the complex number $z = e\, e^{i\varpi}$, and the quasi-periodic approximation given as $z = \sum_{k=1}^{26} b_k\, e^{i(\mu_k t + \varphi_k)}$. The 26 terms μ_k, b_k, φ_k are from table 4, μ_k given in arcseconds per Julian year, and φ_k in degrees. The terms are represented here as the 3 × 26 matrix a_k, b_k, c_k, where a_k is μ_k converted to radians per Julian millennia, b_k is b_k, and c_k is φ_k converted to radians, so that the add-ins function descriptions relate to

$$k + ih = \sum_{k=1}^{26} b_k\, e^{i(a_k t + c_k)}, \text{ and } k + ih = e\, e^{i\varpi}$$

inclination i and longitude of the node Ω are given in Laskar et al (2004) as the complex number $\zeta = \sin(i/2)\, e^{i\Omega}$, and the quasi-periodic approximation given as $\zeta = \sum_{k=1}^{24} a_k\, e^{i(\nu_k t + \phi_k)}$. The 24 terms ν_k, a_k, ϕ_k are from table 5, ν_k given in arcseconds per Julian year, and ϕ_k in degrees. The terms are represented here as the 3 × 24 matrix a_k, b_k, c_k, where a_k is ν_k converted to radians per Julian millennia, b_k is a_k, and c_k is ϕ_k converted to radians, so that the add-ins function descriptions relate to

$$q + ip = \sum_{k=1}^{24} b_k\, e^{i(a_k t + c_k)}, \text{ and } q + ip = \sin(i/2)\, e^{i\Omega}$$

(back to functions *A)
(back to functions *M)

References

Laskar, J., Joutel, F., Boudin, F., **1993**, Orbital, precessional, and insolation quantities for the Earth from –20 Myr to +10 Myr, *Astron Astrophys*. **270**, 522–533 (web source)

Laskar, J., Robutel, P., Joutel, F., Gastineau, M., Correia, A., Levrard, B., **2004**, A long term numerical solution for the insolation quantities of the Earth, *Astron Astrophys*. **428**, 261–285 (web source)

The following 2 functions return the longitude of the Sun, in radians, given from:

Bretagnon, Pierre, and Jean-Louis Simon, **1986**, *Planetary Programs and Tables from –4000 to +2800*, Willmann-Bell, ISBN 0-943396-08-5 (web source)

(back to Additional Functions)

U is dynamical time in 10000 Julian years from J2000 (3652500 days). The 3× matrix is l_i, a_i, v_i, on page 38

pptM(U, 3× matrix) the 3× matrix is stored on your spreadsheet and can go either direction (across or down)

ppt(U) the 3× matrix is stored in memory from the add-in, so that you only need to pass it U

(back to top)

degree 7 Lagrange interpolation polynomial

The following formula is the basis for the data interpolation of k, h, q, p, ψ, ε from page 32 given by

Meeus, Jean, **1998**, *Astronomical Algorithms 2nd ed.*, Willmann-Bell, ISBN 0-943396-61-1 (web source)

Int(t) is the integer less than or equal to t
$x = (t - \text{Int}(t)) + 4$

$$\text{functions } *D = \text{data@}(\text{Int}(t) - 3)\frac{x^7 - 35x^6 + 511x^5 - 4025x^4 + 18424x^3 - 48860x^2 + 69264x - 40320}{-5040}$$

$$+ \text{data@}(\text{Int}(t) - 2)\frac{x^7 - 34x^6 + 478x^5 - 3580x^4 + 15289x^3 - 36706x^2 + 44712x - 20160}{720}$$

$$+ \text{data@}(\text{Int}(t) - 1)\frac{x^7 - 33x^6 + 447x^5 - 3195x^4 + 12864x^3 - 28692x^2 + 32048x - 13440}{-240}$$

$$+ \text{data@}(\text{Int}(t) + 0)\frac{x^7 - 32x^6 + 418x^5 - 2864x^4 + 10993x^3 - 23312x^2 + 24876x - 10080}{144}$$

$$+ \text{data@}(\text{Int}(t) + 1)\frac{x^7 - 31x^6 + 391x^5 - 2581x^4 + 9544x^3 - 19564x^2 + 20304x - 8064}{-144}$$

$$+ \text{data@}(\text{Int}(t) + 2)\frac{x^7 - 30x^6 + 366x^5 - 2340x^4 + 8409x^3 - 16830x^2 + 17144x - 6720}{240}$$

$$+ \text{data@}(\text{Int}(t) + 3)\frac{x^7 - 29x^6 + 343x^5 - 2135x^4 + 7504x^3 - 14756x^2 + 14832x - 5760}{-720}$$

$$+ \text{data@}(\text{Int}(t) + 4)\frac{x^7 - 28x^6 + 322x^5 - 1960x^4 + 6769x^3 - 13132x^2 + 13068x - 5040}{5040}$$

(back to Data functions)

The following is the degree 7 formula's derivative, used for the data interpolation of $\frac{dk}{dt}$, $\frac{dh}{dt}$, $\frac{dq}{dt}$, $\frac{dp}{dt}$, $\frac{d\psi}{dt}$, $\frac{d\varepsilon}{dt}$

$$\text{functions } D*D = \text{data@}(\text{Int}(t) - 3)\frac{7x^6 - 210x^5 + 2555x^4 - 16100x^3 + 55272x^2 - 97720x + 69264}{-5040}$$

$$+ \text{data@}(\text{Int}(t) - 2)\frac{7x^6 - 204x^5 + 2390x^4 - 14320x^3 + 45867x^2 - 73412x + 44712}{720}$$

$$+ \text{data@}(\text{Int}(t) - 1)\frac{7x^6 - 198x^5 + 2235x^4 - 12780x^3 + 38592x^2 - 57384x + 32048}{-240}$$

$$+ \text{data@}(\text{Int}(t) + 0)\frac{7x^6 - 192x^5 + 2090x^4 - 11456x^3 + 32979x^2 - 46624x + 24876}{144}$$

$$+ \text{data@}(\text{Int}(t) + 1)\frac{7x^6 - 186x^5 + 1955x^4 - 10324x^3 + 28632x^2 - 39128x + 20304}{-144}$$

$$+ \text{data@}(\text{Int}(t) + 2)\frac{7x^6 - 180x^5 + 1830x^4 - 9360x^3 + 25227x^2 - 33660x + 17144}{240}$$

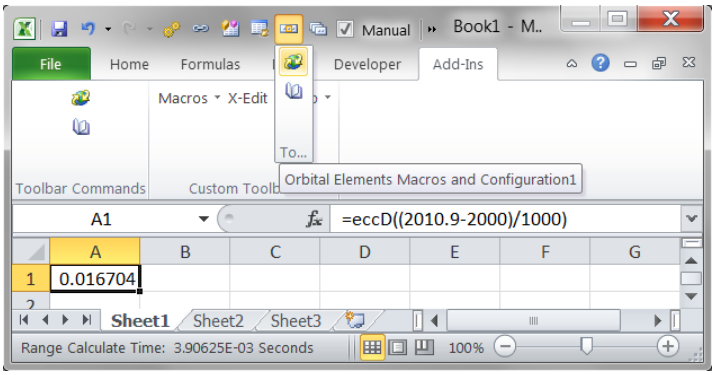
$$+ \text{data@}(\text{Int}(t) + 3)\frac{7x^6 - 174x^5 + 1715x^4 - 8540x^3 + 22512x^2 - 29512x + 14832}{-720}$$

$$+ \text{data@}(\text{Int}(t) + 4)\frac{7x^6 - 168x^5 + 1610x^4 - 7840x^3 + 20307x^2 - 26264x + 13068}{5040}$$

(back to Data Derivatives)

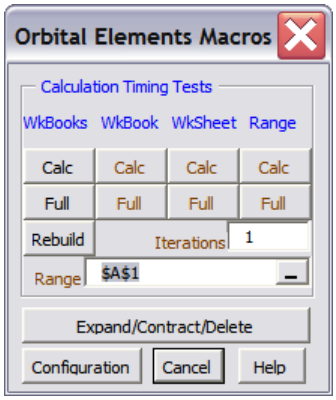
User Form Macros

The Spinning Earth icon in the Add-Ins tab →



↑ lower status bar

provides some calculation timing tests:

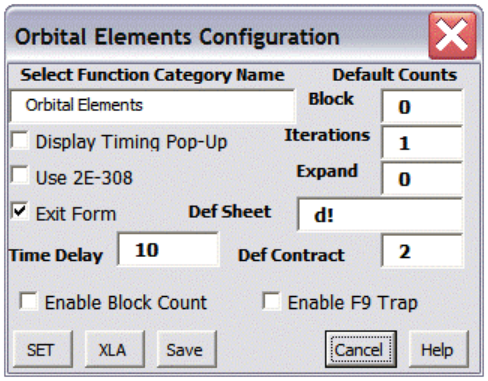


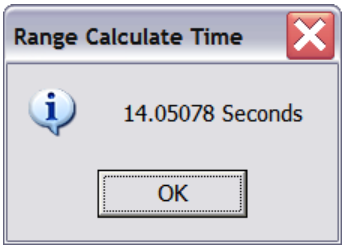
WkBooks Calc is the standard F9 (calculates changes in all open workbooks)
WkBooks Full is Ctrl+Alt+F9 (re-calculates everything in all open workbooks)
WkBooks Rebuild is Ctrl+Alt+Shift+F9 (also rebuilds the dependency table)
WkBook Calc calculates changes in the active workbook
WkBook Full re-calculates everything in the active workbook
WkSheet Calc is Shift+F9 (calculates changes in the active worksheet)
WkSheet Full re-calculates everything in the active worksheet
Range Calc calculates changes to the selected range in the active worksheet
Range Full re-calculates the selected range in the active worksheet

The box labeled "Range" allows you to re-select a range

The box labeled "Iterations" allows you to calculate the same thing multiple times (for timing tests).
Setting this to 0 (zero) allows you to select a "Range" of one cell, and it continues iterating until that cell shows a value *other* than zero.

The "Configuration" button displays the Orbital Elements Configuration form,
which has a checkbox to "Display Timing Pop-Up":





Un-checking this box will not display the pop-up form, but the time can be seen on the lower status bar for 10 seconds. The number of seconds to display it there can be changed in the "Time Delay" box.

"Select Function Category Name" allows you to change the name "Orbital Elements" that appears in Excel's function manager under "category". This only pertains to Office 2007 and later.

"Iterations" box allows you to change the default number in the Timing Tests form (currently set to 1).

"Exit Form" checkbox closes the Timing Tests form after one timing test, otherwise it remains open.

"Use 2E−308" checkbox will direct the data functions to return 2.2251E−308 instead of #NULL! when values of t are outside the range of validity (−20M to +10M).

"Enable F9 Trap" checkbox traps the following keyboard strokes and calculates with these VBA commands instead:

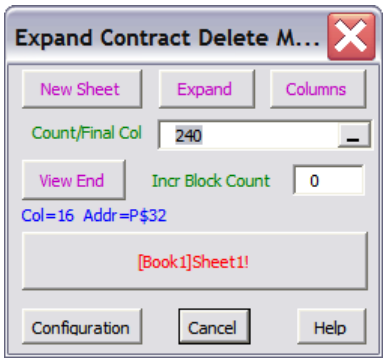
F9 (Calculate) uses Application.Calculate (*much* faster in Excel 2007 and recommended by Microsoft)
Shift+F9 (Calculate Sheet) uses Application.Worksheets(ActiveSheet.Name).Calculate
Ctrl+Alt+F9 (Calculate Full) uses Application.CalculateFull
Ctrl+Alt+Shift+F9 uses Application.CalculateFullRebuild (also rebuilds the dependency table)

"Set" button will set your new configuration options temporarily (until you change them or close Excel).

A standard Left-Click on "Save" will save your settings to a file OE_Cfg.ini in the installation folder so that they will be the same the next time you open Excel. Caution in Excel 97 & 2000: a Right-Click will still resave the Add-In.

The "XLA" button creates another add-in from Excel 2007 and later, OE.xla, compatible with Excel 97-2003.

The "Expand/Contract/Delete" button will open the "Expand Contract or Delete" macro form. The purpose of this macro is to provide a quick and easy method to create more (or less) rows (or columns) of data, and to re-define the data series in your charts by "moving" the last row (or column) of data. It will also rebuild arrays.



The center button at the top toggles between the choices "Expand", "Contract" or "Delete". The button to the right toggles between "Rows" or "Columns".

"Expanding" moves your entire last row down (or last column to the right) to a new location designated in the box labeled "Count/Final Row" (or "Col"). Entering a number moves it that number of rows (or columns). Entering a cell address moves it to the row (or column) containing that cell address. The macro then copies the end row (or column) and pastes to all of the rows (or columns) in between.

"Contracting" moves your last row up (or last column to the left) to the row (or column) containing the cell address you designate in the "Final Row" (or "Final Column") box. It then deletes all of the rows (or columns) beyond.

"Deleting" deletes all of the rows (or columns) from the "Starting Row" (or "Column") and beyond.

The "View End" button will show you the last cell in your last row and last column, or your actual "used range": also displayed below the button (in blue) with both the column number (Col =) and the address (Addr =). This cell address contains the row or column that *will get moved* when you run the macro by clicking on the large button towards the bottom, with the [Workbookname]Sheetname! displayed in red.

To run this macro in a different sheet or in a different workbook: select the sheet and then click "New Sheet" button. Make sure the correct [Workbookname]Sheetname! is displayed, and that the correct address (in blue) in the row or column you wish to "move" is also displayed.

The box labeled "Incr Block Count" will not appear unless the "Enable Block Count" checkbox in the configuration form is checked. This allows you to choose a specific number of rows or columns to paste (in "Expand") or to delete (in "Contract" or "Delete") at one time. For example, if you have 1000 rows of data and expand 4000 columns, using a block count of 500 pastes "blocks" of 500 columns (8 times).

Using too large of a block count number will cause a failure with too much data, resulting in an unusable spreadsheet, especially when contracting or deleting (a current "bug" in Excel 2007 that will hopefully get resolved in the near future). Entering 0 (zero) allows the macro to calculate a "safe" number of rows or columns to paste or delete at once.

As with all macros, "undo" becomes unavailable afterwards. It is highly recommended to "save" before expanding, so that you can simply close the workbook and *don't* "save the changes", instead of contracting (which can be quite slow with huge amounts of data). This is especially recommended if you "Enable Block Count" and get a failure due to too much data, if you choose too large of an "Incr Block Count".

In the Configuration form there are some more options for "Expand/Contract/Delete" under "Default Counts":

"Block" is currently set to 0, so that 0 appears in the "Incr Block Count" box when the form is opened.

"Expand" is currently set to 0, which assumes you want to expand to row 256 or column IV, and will place the appropriate number in the "Count/Final Row" (or Col) box. Set it to 1000 and that will be the number that appears in the "Count/Final Row" box when the form is opened.

"Def Sheet" is currently set to d!, so that if a sheet named "d" exists in the active workbook, that will be the sheet that will appear on the button as [Workbookname]d! when the form is opened, even from another sheet.

"Def Contract" is currently set to 2, so that an appropriate cell address will appear in the "Final Row" (Column) box when contracting with at least one "array" formula on your sheet (an array cannot contract to less than 2).

Additional Functions

The add-in also contains the following functions:

CreateIncrs(t_1, t_j) array function creates even increments of t from t_1 to t_j

LastColumn( ) returns the column number of the last column used on the sheet

LastRow( ) returns the row number of the last row used on the sheet

So that ADDRESS(LastRow( ), LastColumn( )) returns the cell address of the actual used range on the sheet

ChkBinMats(M1,M2,M3,M4)

Function returns TRUE or FALSE. Checks if the 3× matrices on your spreadsheet match the matrices stored in memory from the file LASKARAW.BIN (included in OE.xla*.zip)

M1 is the 3 × 26 matrix of eccentricity variables μ_k, b_k, φ_k, from Laskar et al (2004)

M2 is the 3 × 34 matrix of precession variables ν_k, B_k, ψ_k, from Laskar et al (1993)

M3 is the 3 × 24 matrix of inclination variables ν_k, a_k, ϕ_k, from Laskar et al (2004)

M4 is the 3 × 50 matrix of terms l_i, a_i, v_i, for longitude of Sun from Bretagnon and Simon (1986)

The 3× matrices in LASKARAW. BIN have been converted to radians with "correct" mathematics, rather than the "double precision" in Excel cells (and most computers), and then converted to the closest "double" available. In other words, ChkBinMats will return FALSE if the matrices on your spreadsheet are converted with standard Excel. If you would rather the doubles in LASKARAW. BIN to be converted with standard double precision, you can create a new bin file by downloading CreateBinFile.xls, "paste" new values into the appropriate locations, and run the macro "Create_Raw_Data_File", which will create a file named LASKARAW. BIN in the same folder as CreateBinFile.xls

There is also a subroutine "OEsetup" that when run from the included file OE.csv will re-define the text in the add-in that gets displayed in Excels function manager (from the *fx* button). The text is in the 4th column, "D", of OE.csv. To run this subroutine, open Excels "Macros" form, type in "OEsetup", and then click "Run".

- ☑ It is very important to note that there is currently a "bug" in some versions of Excel that does not allow you to "Change Source" of an Add-In (when you have *alot* of data), under Edit Links, unless Workbook Calculation is set to Automatic.
- ☑ Version 1.7 has a bug fix for saving the configuration settings in Excel 97 and 2000. It no longer resaves the Add-In with a standard Left-Click on the Configuration Screen's "Save" button. Instead it creates a file OE_Cfg.ini in the installation folder. Caution: Right-Clicking "Save" now resaves the Add-In.

For an example of the use of some of these functions, and the 3× matrices to copy and paste, download one of my astronomy spreadsheets ECT*.xls* at http://www.thetropicalevents.com

Hope you enjoy this add-in as much as I do!

For questions or comments email to steve@thetropicalevents.com

Installation notes

This Add-In requires 3 files, make sure they are unblocked: right-click>Properties>General tab>Unblock>Apply

OE.xlam (or OE.xla)
LASKARAW.BIN
OE.chm

Place these files in any folder you choose. I create a folder called "xn" on drive C so the address is very short:

C:\xn\OE.xla*

OE.csv can be placed in the same folder, and when this add-in is also in the same folder as XN.xlam (or XN.xla), the functions will appear in the function manager of Xnumbers, under the category "Orbital Elements".

For Office 2007 and 2010, follow the usual procedure for installing Excel Add-Ins:

1) open Excel
2) from the Office Button (2007) or the File tab (2010), select "Excel Options" , "Add-Ins" , "Go"
3) then "Browse" for and select "OE.xlam"

Go to "Excel Options" from the Office Button or the File tab, click "Trust Center" , "Trust Center Settings" , "Macro Settings" , then check "Enable all macros" and "Trust access to the VBA project". It is also helpful to make sure the add-in is in a Trusted Location in the "Trust Center Settings" under "Trusted Locations". If not: "Add new location" , and type in the address or "Browse" for the location of the appropriate folder. You also might get a "Security Warning" in the toolbar when you open a workbook: Click "Options" and "Enable this content".
It also might be helpful to "Enable automatic update…" in the "Trust Center Settings" under "External Content".

For Office 97-2003, follow the usual procedure for installing Excel Add-Ins:

1) open Excel
2) select "Tools" , "Add-Ins" , "Browse"
3) search for and select "OE.xla"

Make sure to check the box labeled "Trust Access to Visual Basic Project" , or set to "Low" or "Medium" to enable the Add-In, under "Tools" , "Options" , "Security" , "Macro Security".

Licence/Disclaimer