Nonlinear Pendulum

Python-Mathematica code conversion table

Python	**Mathematica**
scipy.special.ellipk(k**2)	EllipticK[k^2]
scipy.special.ellipk(psi, k**2)	EllipticF[psi, k^2]
def T(phi0): return 4*scipy.special.ellipk(np.sin(phi0/2)**2)	T[phi0_] = 4 EllipticK[Sin[phiO/2]^2]
phis = np.linspace(0, np.pi) ts = T(phis) plt.plot(phis, ts) plt.ylim(0, 30)	Plot[T[phiO], {phi0,0,Pi}, PlotRange -> {0,30}]
def psi(t, phi0): return scipy.special.ellipj(t, np.sin(phi0/2)**2)[3]	psi[t_, phiO_] = JacobiAmplitude[t, Sin[phi0/2]^2]
def sinepsi(t, phi0): return scipy.special.ellipj(t, np.sin(phi0/2)**2)[0]	sinepsi[t_, phiO_] = JacobiSN[t, Sin[phiO/2]^2]
def phinorm(x, phi0): return 2*np.arcsin(np.sin(phi0/2) * sinepsi(x* T(phi0), phi0))/phi0	phinorm[x_, phiO_] := 2 ArcSin[Sin[phiO/2] sinepsi[x T[phiO], phiO]]/phiO
phi0[0] = 0.1*np.pi	phiO[1] = N[0.1 Pi]
phi0[4] = 0.999*np.pi	phiO[5] = N[0.999 Pi]
flist = [lambda x: phinorm(x, phi0[i]) for i in range(5)]	flist = Table[phinorm[x, phiO[i]], {i, 5}]
np.abs(scipy.fftpack.fft(list_))/np.sqrt(len(list_))	foulist = Abs[Fourier[list]]
m = sympy.Symbol(“m”) f = 1/sympy.sqrt(1 – m*sympy.sin(psi**2)) f.series(x, point=0, n=10)	f = 1/Sqrt[1 – m Sin[psi]^2] g = Series[f, {m, 0, 10}]
.subs({m: Sin[phiO/2]^2})	/. m -> Sin[phiO/2]^2