temp = MathTimeLib.globals.mu / (MathTimeLib.globals.re \* MathTimeLib.globals.re);

// reverse order of summations to get it right!!

for (m = 0; m <= order; m++)

{

for (L = m; L <= order; L++)

{

if (normalized == 'n') // unnormalized

{

if (m == 0)

{

aPertM[0] = aPertM[0] + temp \* (-gravData.c[L, 0] \* VArr[L + 1, 1]);

aPertM[1] = aPertM[1] + temp \* (-gravData.c[L, 0] \* WArr[L + 1, 1]);

aPertM[2] = aPertM[2] + temp \* (L + 1) \* (-gravData.c[L, 0] \* VArr[L + 1, 0]);

}

else

{

// double temp1 = MathTimeLibr.factorial(L - m + 2) / MathTimeLibr.factorial(L - m);

// 1975 Giacaglia pg 6 shows the conversion, more stable

double temp1 = temp \* 0.5 \* (L - m + 1) \* (L - m + 2);

aPertM[0] = aPertM[0] + 0.5 \* temp \* (-gravData.c[L, m] \* VArr[L + 1, m + 1] - gravData.s[L, m] \* WArr[L + 1, m + 1]) +

temp1 \* (gravData.c[L, m] \* VArr[L + 1, m - 1] + gravData.s[L, m] \* WArr[L + 1, m - 1]);

aPertM[1] = aPertM[1] + 0.5 \* temp \* (-gravData.c[L, m] \* WArr[L + 1, m + 1] + gravData.s[L, m] \* VArr[L + 1, m + 1]) +

temp1 \* (-gravData.c[L, m] \* WArr[L + 1, m - 1] + gravData.s[L, m] \* VArr[L + 1, m - 1]);

aPertM[2] = aPertM[2] + temp \* ((L - m + 1) \* (-gravData.c[L, m] \* VArr[L + 1, m] - gravData.s[L, m] \* WArr[L + 1, m]));

}

}

else // normalized

{

// note Varr and WArr not formulated for normalized yet

if (m == 0)

{

aPertM[0] = aPertM[0] + temp \* (-gravData.cNor[L, 0] \* VArr[L + 1, 1]);

aPertM[1] = aPertM[1] + temp \* (-gravData.cNor[L, 0] \* WArr[L + 1, 1]);

aPertM[2] = aPertM[2] + temp \* (L + 1) \* (-gravData.cNor[L, 0] \* VArr[L + 1, 0]);

}

else

{

// double temp1 = MathTimeLibr.factorial(L - m + 2) / MathTimeLibr.factorial(L - m);

// 1975 Giacaglia pg 6 shows the conversion, more stable

double temp1 = temp \* 0.5 \* (L - m + 1) \* (L - m + 2);

aPertM[0] = aPertM[0] + 0.5 \* temp \* (-gravData.cNor[L, m] \* VArr[L + 1, m + 1] - gravData.sNor[L, m] \* WArr[L + 1, m + 1]) +

temp1 \* (gravData.cNor[L, m] \* VArr[L + 1, m - 1] + gravData.sNor[L, m] \* WArr[L + 1, m - 1]);

aPertM[1] = aPertM[1] + 0.5 \* temp \* (-gravData.cNor[L, m] \* WArr[L + 1, m + 1] + gravData.sNor[L, m] \* VArr[L + 1, m + 1]) +

temp1 \* (-gravData.cNor[L, m] \* WArr[L + 1, m - 1] + gravData.sNor[L, m] \* VArr[L + 1, m - 1]);

aPertM[2] = aPertM[2] + temp \* ((L - m + 1) \* (-gravData.cNor[L, m] \* VArr[L + 1, m] - gravData.sNor[L, m] \* WArr[L + 1, m]));

}

}

} // for m

} // for L

straccum = straccum + "apertM ef 4 4 " + aPertM[0].ToString() + " " + aPertM[1].ToString() + " " + aPertM[2].ToString() + "\n";

// Calculate accelerations ax,ay,az

ax = ay = az = 0.0;

for (m = 0; m <= order; m++)

for (n = m; n <= order; n++)

if (m == 0)

{

if (normalized == 'n')

C = gravData.c[n, 0]; // = C\_n,0

else

C = gravData.cNor[n, 0]; // = C\_n,0

aPertM1[0] -= C \* V[n + 1, 1];

aPertM1[1] -= C \* W[n + 1, 1];

aPertM1[2] -= (n + 1) \* C \* V[n + 1, 0];

}

else

{

if (normalized == 'n')

{

C = gravData.c[n, m]; // = C\_n,m

S = gravData.s[n, m]; // = S\_n,m he stores CS differently m - 1, n

}

else

{

// note V and W not formulated for normalized yet

C = gravData.cNor[n, m]; // = C\_n,m

S = gravData.sNor[n, m]; // = S\_n,m he stores CS differently m - 1, n

}

Fac = 0.5 \* (n - m + 1) \* (n - m + 2);

aPertM1[0] += +0.5 \* (-C \* V[n + 1, m + 1] - S \* W[n + 1, m + 1])

+ Fac \* (+C \* V[n + 1, m - 1] + S \* W[n + 1, m - 1]);

aPertM1[1] += +0.5 \* (-C \* W[n + 1, m + 1] + S \* V[n + 1, m + 1])

+ Fac \* (-C \* W[n + 1, m - 1] + S \* V[n + 1, m - 1]);

aPertM1[2] += (n - m + 1) \* (-C \* V[n + 1, m] - S \* W[n + 1, m]);

}

// Body-fixed acceleration

// now get correction for initial V, W formulation

aPertM1[0] = (MathTimeLib.globals.mu / (MathTimeLib.globals.re \* MathTimeLib.globals.re)) \* aPertM1[0];

aPertM1[1] = (MathTimeLib.globals.mu / (MathTimeLib.globals.re \* MathTimeLib.globals.re)) \* aPertM1[1];

aPertM1[2] = (MathTimeLib.globals.mu / (MathTimeLib.globals.re \* MathTimeLib.globals.re)) \* aPertM1[2];

// ----------Partial derivatives of disturbing potential ------- }

double magr = MathTimeLibr.mag(recef);

ror = MathTimeLib.globals.re / magr;

dRdr = 0.0;

dRdlat = 0.0;

dRdlon = 0.0;

temp = ror;

// sum the Legendre polynomials for the given order

for (L = 2; L <= order; L++)

{

// will do the power as each L is indexed }

temp = temp \* ror;

sumM1 = 0.0;

sumM2 = 0.0;

sumM3 = 0.0;

for (m = 0; m <= L; m++)

{

// unnormlaized should sum in reverse to preserve accuracy

if (normalized == 'n') // unnormalized

{

sumM1 = sumM1 + legarrGU[L, m] \* (gravData.c[L, m] \* trigArr[m, 1] + gravData.s[L, m] \* trigArr[m, 0]);

// no m + 1????????, no I think it IS +1

sumM2 = sumM2 + (legarrGU[L + 1, m] - trigArr[m, 2] \* legarrGU[L, m]) \*

(gravData.c[L, m] \* trigArr[m, 1] + gravData.s[L, m] \* trigArr[m, 0]);

sumM3 = sumM3 + legarrGU[L, m] \* m \* (gravData.s[L, m] \* trigArr[m, 1] - gravData.c[L, m] \* trigArr[m, 0]);

}

else // normalized

{

sumM1 = sumM1 + legarrGN[L, m] \* (gravData.cNor[L, m] \* trigArr[m, 1] + gravData.sNor[L, m] \* trigArr[m, 0]);

sumM2 = sumM2 + (legarrGN[L + 1, m] - trigArr[m, 2] \* legarrGN[L, m]) \*

(gravData.cNor[L, m] \* trigArr[m, 1] + gravData.sNor[L, m] \* trigArr[m, 0]);

sumM3 = sumM3 + legarrGN[L, m] \* m \* (gravData.sNor[L, m] \* trigArr[m, 1] - gravData.cNor[L, m] \* trigArr[m, 0]);

}

} // for m

dRdr = dRdr + temp \* (L + 1) \* sumM1;

dRdlat = dRdlat + temp \* sumM2;

dRdlon = dRdlon + temp \* sumM3;

} // for L

muor = MathTimeLib.globals.mu / magr;

dRdr = -muor / magr \* sumM1;

dRdlat = muor \* sumM2;

dRdlon = muor \* sumM3;

// ----------Non - spherical perturbative acceleration ------------ }

oor = 1.0 / magr;

RDelta = recef[0] \* recef[0] + recef[1] \* recef[1];

double oordeltasqrt = 1.0 / Math.Sqrt(RDelta);

oordelta = 1.0 / RDelta;

temp = oor \* dRdr - recef[2] \* oor \* oor \* oordeltasqrt \* dRdlat;

//double tmp = MathTimeLib.globals.mu / (Math.Pow(MathTimeLibr.mag(recef), 3));

aPertG[0] = temp \* recef[0] - oordelta \* dRdlon \* recef[1]; // - tmp \* recef[0];

aPertG[1] = temp \* recef[1] + oordelta \* dRdlon \* recef[0]; // - tmp \* recef[1];

aPertG[2] = oor \* dRdr \* recef[2] + oor \* oor \* Math.Sqrt(RDelta) \* dRdlat; // - tmp \* recef[2];

straccum = straccum + "apertG ef 4 4 " + aPertG[0].ToString() + " " + aPertG[1].ToString() + " " + aPertG[2].ToString() + "\n";