1 Die Klasse Vector3D

Name der Methode	Kurzbeschreibung
Vector3D()	Erzeugt den Nullvektor (0,0,0)
Vector3D (double x, double y, double z)	double-Konstruktor
Vector3D(const Vector3D& other)	Der Copy-Konstruktor
Vector3D (double* arr)	Array-Konstruktor
Vector3D vecAdd(const Vector3D& other)	Vektoraddition
Vector3D vecSub(const Vector3D& other)	Vektorsubtraktion
Vector3D scale(double factor)	S-Multiplikation
Vector3D cross(const Vector3D& right)	Kreuzprodukt
double dot(const Vector3D& other)	Skalarprodukt
double getLength()	Laenge eines Vektors
Vector3D normalize()	Normlisierung
bool isNormalized()	Prüft, ob Vektor normalisiert
bool equals(const Vector3D& other)	Prüft auf Gleichheit
double getAngle(Vector3D& other)	Liefert Winkel zwischen den Vektoren
bool isOrthogonal(const Vector3D& other)	Prüft auf orthogonalität
Polar carToPolar()	Wandelt in r, ϕ, θ
Vector3D polarToCar()	Wandelt in x, y, z
Vector3D matVecMult(Matrix3D& M)	Matrix-Vektor-Multiplikation
Vector3D qRotate(Quaternion& q)	Rotation mit einem Quaternion
Vector3D mRotate(Matrix3D& M)	Rotation mit einer Matrix
Vector3D vecRotate(AngleAxis& u_phi)	Rotation mit Achse und Winkel
Vector4D to4D()	zu Vector4D
void print()	Konsolenausgabe

2 Die Klasse Vector4D

Vector4D()	Erzeugt den Vector4D (0,0,0,1)
Vector4D (double x, double y, double z,double scale)	double-Konstruktor
Vector4D(const Vector4D& other)	Copy-Konstruktor
Vector4D (double* arr)	Array-Konstruktor
Vector4D matVecMult(const Matrix4D& M)	Matrix-Vektor-Multiplikation
Vector3D to3D()	als Vector3D darstellen
Vector4D mRotate(const Matrix4D& M)	Rotation+Translation mit einer Matrix4D
void print()	Konsolenausgabe

3 Die Klasse YPR

YPR()	Es wird ein YPR (0,0,0) erzeugt
YPR(const YPR& other)	Copy-Konstruktor
YPR(double yaw,double pitch,double roll)	double-Konstruktor
YPR(Quaternion q)	Quaternion-Konstruktor
YPR(Matrix3D& M)	Matrix-Konstruktor
YPR(AngleAxis& other)	AngleAxis Konstruktor
Matrix3D toMatrix3D()	YPR zu Matrix3D
Quaternion toQuaternion()	YPR zu Quaternion
AngleAxis toAngleAxis()	YPR zu AngleAxis
void print()	Konsolenausgabe

4 Die Klasse Quaternion

Quaternion()	Erstellt die Identität q= (1,0,0,0)
Quaternion(const Quaternion& q)	Copy-Konstruktor
Quaternion(double q0,double q1, double q2, double q3)	double-Konstruktor
Quaternion(double* arr)	Array-Konstruktor
Quaternion(double q0,const Vector3D& q)	Skalar-Vektor-Konstruktor
Quaternion(AngleAxis& other)	Angle-Axis-Konstruktor
Quaternion(Matrix3D& other)	Matrix-Konstruktor
Quaternion(YPR& other)	YPR-Konstruktor
double getAngle()	Liefert Rotationswinkel
Vector3D getVec()	Liefert Rotationsachse
Quaternion qAdd(const Quaternion& other)	Quaternionaddition
Quaternion qSub(const Quaternion& other)	Quaternionsubtraktion
Quaternion scale(double factor)	S-Multiplikation
Quaternion qMult(const Quaternion& right)	Quaternionmultiplikation
Quaternion conjugate()	komplex-konjugiertes Quaternion
Quaternion invert()	inverses Quaterion
Quaternion qDivide(const Quaternion& denominator)	Quaterniondivision
double getNorm()	Liefert den Betrag des Quaternion
Quaternion normalize()	Normalisiert das Quaternion
bool isNormalized()	Prüft auf Normalisierung
bool equals(const Quaternion& other)	Prüft auf Gleichheit
AngleAxis toAngleAxis()	Quaternion zu AngleAxis
Matrix3D toMatrix3D()	Quaternion zu Matrix3D
YPR toYPR()	Quaternion zu YPR
void print()	Konsolenausgabe

5 Die Klasse AngleAxis

AngleAxis()	Erstellt das AngleAxis([0,0,0],0)
AngleAxis(const AngleAxis& other)	Copy-Konstruktor
AngleAxis(Vector3D& u,double phi)	Vektor-Winkel Konstruktor
AngleAxis(Quaternion& q)	Quaternion-Konstruktor
AngleAxis(Matrix3D& M)	Matrix-Konstruktor
AngleAxis(YPR& ypr)	AngleAxis Konstruktor
Quaternion toQuaternion()	AngleAxis zu Quaternion
Matrix3D toMatrix3D()	AngleAxis zu Matrix3D
YPR toYPR()	AngleAxis zu YPR
void print()	Konsolenausgabe

6 Die Klasse Matrix4D

Matrix4D()	Erstellt die Identität
Matrix4D(Matrix3D& rot,Vector3D trans)	Matrix3D-Vector3D Konstruktor
Matrix4D(double* arr)	Array-Konstruktor
Matrix4D(const Matrix4D& other)	Copy-Konstruktor
Matrix3D getRotation()	Liefert Rotationsmatrix
Vector3D getTranslation()	Liefert Translation
Matrix4D scale(double factor)	homogene Matrixskalierung
Matrix4D mMult(const Matrix4D& right)	Matrixmultiplikation
Matrix4D invert()	inversion der Matrix4D
void print()	Konsolenausgabe

7 Die Klasse CoordinateFrame3D

CoordinateFrame3D()	Erstellt ein Frame aus 4 Nullvektoren
CoordinateFrame3D(Vector3D& x, Vector3D& y, Vector3D& z, Vector3D& origin)	4 Vektor-Konstruktor
CoordinateFrame3D(Vector3D& x, Vector3D& y, Vector3D& origin)	TRIAD-Konstruktor
CoordinateFrame3D(const CoordinateFrame3D& other)	Copy-Konstruktor
Matrix4D mapTo(CoordinateFrame3D& other)	Matrix4D von this nach other
CoordinateFrame3D translate(Vector3D& trans)	Verschiebt den Frame
CoordinateFrame3D rotate(Matrix3D& rot)	Rotiert den Frame mit einer Matrix
CoordinateFrame3D rotate(Quaternion& q)	Rotiert den Frame mit einem Quaternion

8 Die Klasse TwoLineElement

TwoLineElement(string* twoLine)	Erstellt aus Stringarray ein TLE

9 Die Klasse ComputationModel

ComputationModel(TwoLineElement tle)	TLE-Konstruktor
double* sgp4(double elapsedTime, TwoLineElement tle)	Berechnet Position nach SGP4

10 Die Klasse Polar

Polar()	Erstellt ein Polar (0,0,0)
Polar(double r,double phi,double theta)	double-Konstruktor
Polar(const Polar& other)	Copy-Konstruktor
Polar(const Vector3D& other)	kartesisch zu Polar-Konstruktor
Vector3D toCartesian()	polar zu kartesisch
void print()	Konsolenausgabe

11 Die Klasse Complex

Complex()	Erstellt die komplexe Zahl z = 0 +i*0
Complex(const Complex& other)	Copy-Konstruktor
Complex(double Re, double Im)	double-Konstruktor
Complex cAdd(const Complex& other)	komplexe Addition
Complex cSub(const Complex& other)	komplexe Subtraktion
Complex cScale(double scale)	S-Multiplikation
Complex cMult(const Complex& other)	komplexe Multiplikation
Complex cPow(int exponent)	komplexe Potenzierung
Complex cExp()	komplexe Exponentialfunktion

12 Globale Funktionen

Vector3D rotateX(Vector3D& s,double angle)	Rotiert s um die x-Achse
Vector3D rotateY(Vector3D& s,double angle)	Rotiert s um die y-Achse
Vector3D rotateZ(Vector3D& s,double angle)	Rotiert s um die z-Achse
double daysSinceY2k(int year,int month,int day, int hour,int minute,double second)	Errechnet vergangene Tage seit Jahr 2000
Matrix3D eciToECEF(double days2k)	Liefert Rotationsmatrix für ECEF in ECI Koordinaten
Vector3D geodeticToECEF(Vector3D& other)	Transformiert other von ECEF in in geodätische Koordinaten
Vector3D ecfToGeodetic(Vector3D& other)	Wandelt other von geodätischen in ECEF Koordinaten
void FFT(Complex* a, int n, int lo)	FFT eines komplexen Vektors
void IFFT(Complex* a, int n, int lo)	inverse FFT eines komplexen Vektors

13 Die Klasse Matrix3D

Matrix3D()	Erstellt die Identität
Matrix3D(const Vector3D& coloumn1, const Vector3D& coloumn2,const Vector3D& coloumn3)	Erstellt Matrix3D aus 3 Spaltenvektoren
Matrix3D(double* arr)	array-Konstruktor
Matrix3D(const Matrix3D& other)	Copy-Konstruktor
Matrix3D(const Vector3D& init)	Diagonalmatrix aus Vector3D
Matrix3D(YPR& other)	YPR-Konstruktor
Matrix3D(AngleAxis& other)	AngleAxis-Konstruktor
Matrix3D(Quaternion& other)	Quaternion-Konstruktor
Vector3D getVec()	Liefert die Rotationsachse
double getAngle()	Liefert den Rotationswinkel
Vector3D getRow1()	Liefert erste Zeile
Vector3D getRow2()	Liefert zweite Zeile
Vector3D getRow3()	Liefert dritte Zeile
Vector3D getColoumn1()	Liefert erste Spalte
Vector3D getColoumn2()	Liefert zweite Spalte
Vector3D getColoumn3()	Liefert dritte Spalte
void setRow1(Vector3D& row)	setzt erste Zeile
void setRow2(Vector3D& row)	setzt zweite Zeile
void setRow3(Vector3D& row)	setzt dritte Zeile
void setColoumn1(Vector3D& coloumn)	setzt erste Spalte
void setColoumn2(Vector3D& coloumn)	setzt zweite Spalte
void setColoumn3(Vector3D& coloumn)	setzt dritte Spalte
Matrix3D mAdd(const Matrix3D& other)	Matrixaddition
Matrix3D mSub(const Matrix3D& other)	Matrixsubtraktion
Matrix3D scale(double factor)	Skalierung
Matrix3D mMult(const Matrix3D& right)	Matrixmultiplikation
Matrix3D cofac()	Kofaktormatrix
Matrix3D adjoint()	Adjunkte Matrix
Matrix3D invert()	inverse Matrix
Matrix3D transpose()	Transponierte Matrix
Matrix3D mDivide(Matrix3D& denominator)	Matrixdivision
Matrix3D rotateX(double angle)	Liefert Fundamentalmatrix um x
Matrix3D rotateY(double angle)	Liefert Fundamentalmatrix um y
Matrix3D rotateZ(double angle)	Liefert Fundamentalmatrix um z
double determinant()	Berechnet Determinante
bool isOrthogonal()	Prüft auf orthogonalität
bool equals(const Matrix3D& other)	Prüft auf Gleichheit
Quaternion toQuaternion()	Matrix3D zu Quaternion
YPR toYPR()	Matrix3D zu YPR
AngleAxis toAngleAxis()	Matrix3D zu AngleAxis
void print()	Konsolenausgabe