Rotationen

Vorüberlegungen und Pseudo-Code

1.) Viele Rechnungen mit Vektoren, daher Wunsch: struct ändern, neu: Mit Vektoren:

```
typedef double[3] Vector;       // 3 dimensionaler Vektor

typedef double[3][3] Matrix;    // 3 x 3 Matrix

typedef struct
{
  int id;
  Vector x;                     // x[0], x[1], x[2] an Stelle von x, y, z
                                // oder geeignet umbenennen: Vector pos;
  char *element_symbol;
} Atom;
```

2.) Beschreibung der zu erstellenden Funktion: Diese Funktion rotiert Teile eines Moleküls **m** um eine Achse, gegeben durch eine Bindung **b** (= Verbindungslinie **r** zweier Atome) dieses Moleküls um den Winkel **φ** (gegeben in Radian), mögliche Werte (-2π ... +2π) und berechnet die neuen Koordinaten aller Atome.

```
void molekule_rotate(Molecule m, const Bond b, const double phi)
```

Zusätzlich, evtl. besser zum Testen:

```
Molecule molekule_rotated(const Molecule m, const Bond b, const double phi)
```

3.) Pseudocode:

3.1) Verschiebe den Koordinatenursprung in das erste der zwei Bindungsatome:

```
For each Atom a in Molekule m do
a.x = vector_substract(a.x,
      b.first.x)
```

3.2) Bestimme den Winkel , den die Verbindungslinie r der beiden Atome mit der jetzigen z-Achse bildet gemäß

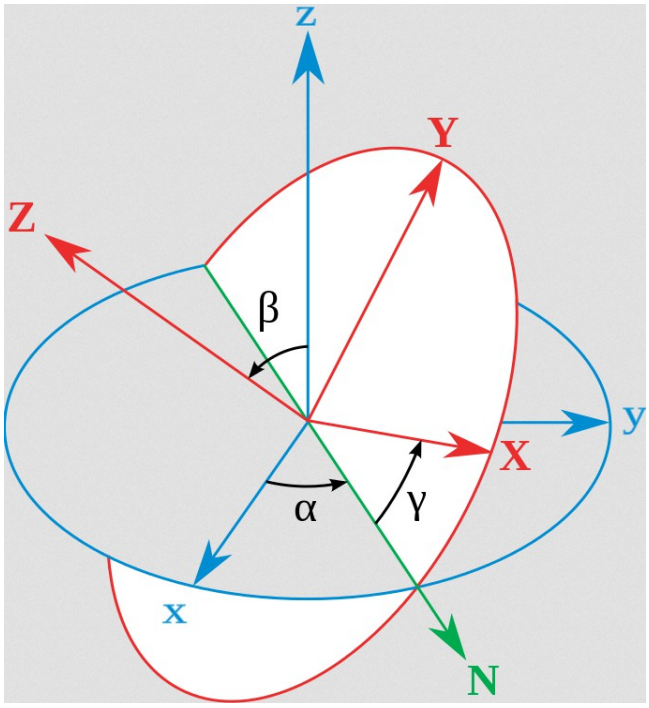
$$\vec{r} = \overrightarrow{b.last} - \overrightarrow{b.first} = \overrightarrow{b.last} - 0$$

$$\vartheta = \arccos\left(\frac{b.lastx.x[2]}{\sqrt{(\vec{r}\cdot\vec{r})}}\right)$$

Damit sind die Eulerschen Winkel[1] für die folgende Rotation definiert:

```
e1 = 0; e2 = theta, e3 = 0;
```

Mit dieser Wahl bleibt die x-Achse erhalten.



3.3) Berechne die Rotationsmatritzen R und R^{-1}. (Da die Drehmatrix orthogonal ist, ist $R^{-1} = R^T$).

```
Matrix Rot, Rot_Inv;
double e1, e2, e3;
Rot = euler_rotate(e1, e2, e3);
Rot_Inv = matrix_transpose(R);
```

1 http://de.wikipedia.org/wiki/Eulersche_Winkel

3.4) Berechne die Koordinaten im gedrehten Koordinatensystem:

```
For each Atom a in Molekule m do
a.x = matrix_times_vector(Rot, a.x)
```

3.5) Drehe **nur die Atome oberhalb der Bindungslini**e um den gewünschten Winkel φ um die (neue) z-Achse:

```
For each Atom a in Molekule m.right do
a.x =  vector_rotate_z(a.x, phi);
```

3.6.) Jetzt muss die Drehung des Koordinatensystems wieder rückgängig gemacht werden:

```
For each Atom a in Molekule m do
a.x = matrix_times_vector(Rot_Inv, a.x)
```

3.7.) Und schließlich muss auch die Verschiebung aus 3.1 wieder rückgängig gemacht werden:

```
For each Atom a in Molekule m do
a.x = vector_add(a.x, b.first.x)
```

4.) Kontrolle

Bei dem ganzen Schieben und Drehen kann es einem leicht schwindelig werden. Daher sollte man testen, ob bei einer Drehung um $\varphi = 0$ oder $\varphi = 2\pi$ auch wirklich die ursprüngliche Konfiguration wieder herauskommt. Da wir mit Gleitkommazahlen rechnen, werden die Koordinaten nicht exakt übereinstimmen. Daher sollte man die Größe

$$\delta = \sum_{atome} \sum_{i=0}^{2} \left(atom.position[i]_{vorher} - atom.position[i]_{nachher} \right)^2$$

berechnen können, das müsste dann ein sehr kleiner Wert sein. TODO: Geeignete Funktionen dazu schreiben.

Außerdem müsste es möglich sein, mal nur eine einzige große Drehung auszuführen und sich dann das geänderte Molekül anzuschauen.

5.) Hilfsfunktionen, auch für andere Programmteile hilfreich (evtl. separate Datei)

```
Vector vector_add(const Vector x, const Vector y)
// Gibt einen Vektor z = x + y zurück

Vector vector_substract(const Vector x, const Vector y)
// Gibt einen Vektor z = x - y zurück

double vector_skalar_mult(const Vector x, const double a)
// Gibt einen Vektor z = a mal x zurück

double vector_skalarproduct(const Vector x, const Vector y)
// Gibt das Skalarprodukt z = x mal y zurück.

Vector vector_rotate_z(const Vector x, const double phi)
// Gibt den um den Winkel phi um die z-Achse rotierten Vektor x zurück

Vector matrix_times_vector(const Matrix A, const Vector x)
// Gibt das Produkt einer 3x3 Matrix A mit einem Spaltenvektor x zurück

Matrix matrix_transpose(const Matrix A)
// Gibt die zu A transponierte Matrix At zurück
```

```
Matrix euler_rotate(const double e1, const double e2, const double e3)
// Erzeugt eine Drehmatrix in der "x-Konvention"
// aus den 3 Eulerschen Winkeln e1, e2, e3
// Siehe http://de.wikipedia.org/wiki/Eulersche_Winkel
```

$$\mathbf{A} = \begin{pmatrix} \cos\psi\cos\phi - \cos\theta\sin\phi\sin\psi & \cos\psi\sin\phi + \cos\theta\cos\phi\sin\psi & \sin\psi\sin\theta \\ -\sin\psi\cos\phi - \cos\theta\sin\phi\cos\psi & -\sin\psi\sin\phi + \cos\theta\cos\phi\cos\psi & \cos\psi\sin\theta \\ \sin\theta\sin\phi & -\sin\theta\cos\phi & \cos\theta \end{pmatrix}$$

Eventuell weitere Matrix-Generatoren für die Eulerschen Winkel in anderen der y-Konvention oder der xyz-Konvention, siehe Goldstein, Klassische Mechanik. Hier noch das Bild aus dem Goldstein zur x-Konvention:

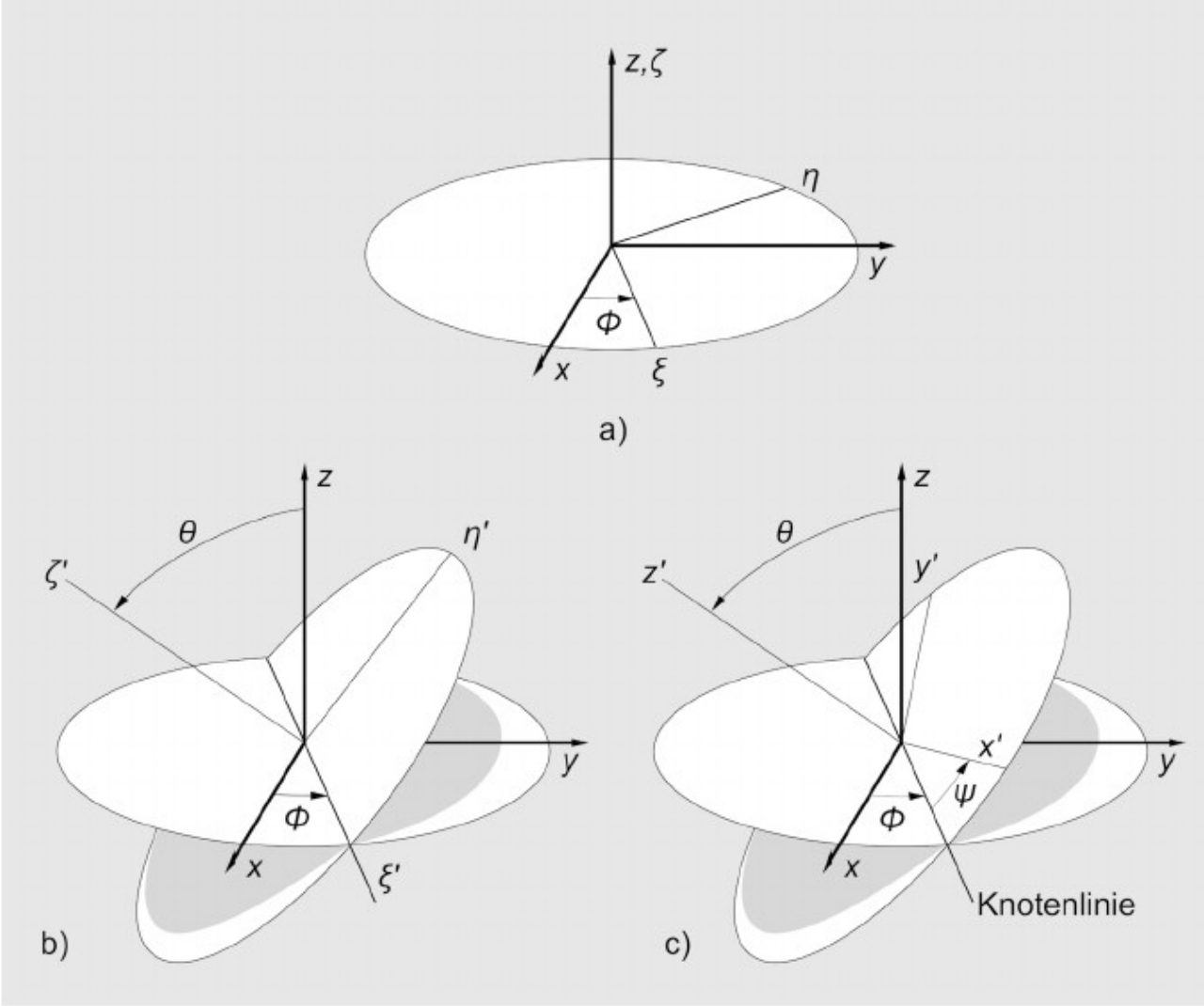


Abbildung 4.7 Die Drehungen, die die Eulerschen Winkel definieren.