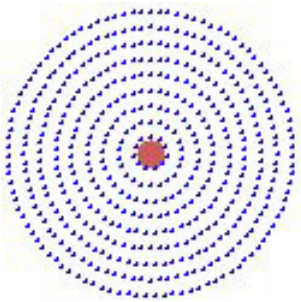
Übungsblatt 7
C/C++ Wechselwirkende Galaxien
(Ausgabe 12.06.2014)

---

1. Aufgabe *Wechselwirkung zweier Galaxien* ****

Wir wollen die Wechselwirkung zweier Galaxien simulieren. Dazu werden wir folgende (vereinfachende) Annahmen machen:

- Jede Galaxie besteht aus einer zentralen Punktmasse, die die gesamte Masse der Galaxie enthält.
- Um die zentrale Punktmasse bewegen sich Testteilchen, anfangs auf *Keplerorbits*. Diese Testeilchen spüren nur das Potenzial der zentralen Punktmassen, die Wechselwirkung der Testteilchen untereinander kann vernachlässigt werden.



Implementieren Sie ein Programm `galaxy`, welches mittels Runge-Kutta-Verfahren die Passage zweier Galaxien simuliert und aus folgenden Funktionen besteht:

- Funktion `main` initialisert `Xgraphics`, ruft `initial` und loopt über der Zeit. In der Schleife werden `integrate` und `output` gerufen.
- Funktion `initial` legt die Startbedingungen und Parameter fest. Es gibt die beiden Massen `m1` und `m2`. Um diese gibt es jeweils `nr` äquidistante Ringe, die jeweils in einer Ebene liegen. Die Anzahl von Testteilchen auf den Ringen nimmt von innen nach außen zu. Die *dreidimensionalen* Positionen und Geschwindigkeiten der Zentralmassen und Testteilchen können in einem Vektor `double x[]` gespeichert werden.

  Die Anfangsgeschwindigkeiten der Zentralmassen zeigen entlang der x-Achse. Der Abstand von der x-Achse (Stoßparameter) zusammen mit einem Inklinationswinkel ergibt die y- und z-Koordinaten. Die Startbedingungen sind für die Galaxien symmetrisch bezüglich $(0; 0; 0)$, aber mittels der Massen skaliert:

```
// Massen:
 m1  = 1. ;
 m2  = 1. ;
// Galaxie m2 :
 r2   = 2. ;
 inc_ang1 = -0.1 * M_PI ;
 xs2  = -4. ;
 ys2  = cos(inc_ang1) * r2 ;
 zs2  = sin(inc_ang1) * r2 ;
 vx2 = 0.1 ; // v(t=0)
 vy2 = 0.0 ;
 vz2 = 0.0 ;
// Galaxie m1
 fac = - m2 / m1 ;
 xs1 = fac * xs2 ;
 ys1 = fac * ys2 ;
 zs1 = fac * zs2 ;
 vx1 = fac * vx2 ;
 vy1 = fac * vy2 ;
 vz1 = fac * vz2 ;
```

Diese Werte werden in den Vektor `x[]` übertragen, also
`x[0] = xs1 ; x[1] = vx1 ; ... ; x[11] = vz2 ;`

Schließlich werden die Startpositionen und Geschwindigkeiten (Keplerkreisbahnen) der Testteilchen festgelegt. Dazu läuft eine Schleife über die Ringe und darin eine weitere Schleife über den Winkel:

```
np = 2 ;                  // die zentralen Massen sind schon in x[]
for ( int i = ... ) {     // Loop ueber Ringe
  rp   = (i+1) * rstep ; // radialer Abstand von Zentralmasse
  vp   = ... ;           // Kreisbahngeschwindigkeit, G=1 (!)
  npx  = ... ;           // Anzahl der Teilchen auf aktuellem Ring
  dphi = 2.*M_PI/double(npx) ; // Winkelbruchteil
  for ( int n = 0 ; ... ) {    // Loop ueber Teilchen auf dem Ring
    wphi = double(n) * dphi ;  // Winkel
    dx = rp * sin(wphi) ;      // x relativ zur Zentralmasse
    dy = ... ;
    vx = vp * cos(wphi) ;      // v relativ zu v_Zentralmasse
    vy = vp * (-1.*sin(wphi)) ;
    n0 = 6 * (np + n) ;        // Index berechnen
    x[n0]   = xs1 + dx ;       // absolute Koordinaten
    x[n0+1] = vx1 + vx ;
     ...
    x[n0+4] = zs1 ;
    x[n0+5] = vz1 ;
  }
  np = np + npx ;
 }
```

Für die andere Galaxie wird ganz analog verfahren, allerdings kommt jetzt noch ein weiterer Inklinationswinkel ins Spiel:

```
...
  dx = rp * sin(wphi) ;
  dy = rp * cos(wphi) * cos(inc_ang2) ;
  dz = rp * cos(wphi) * sin(inc_ang2) ;
  vx = vp * cos(wphi) ; // wie oben
  vy = vp * (-1.*sin(wphi)) * cos(inc_ang2) ;
  vz = vp * (-1.*sin(wphi)) * sin(inc_ang2) ;
...
```

- In der Funktion `integrate` wird das Runge-Kutta-Verfahren angewandt. Die ersten beiden Koeffizienten `k1, k2` (entsprechen den Ableitungen dx/dt) werden mittels der Funktion `deriv` z.B. so berechnet:

```
deriv (xold, k1, ndim, m1, m2) ;
for (int i = 0  ; i < ndim ; i++ )
    xnew[i] = xold[i] + k1[i] * 0.5 * dt ;
deriv (xnew, k2, ndim, m1, m2) ;
 ...
```

- Die Funktion `deriv` liefert die Koeffizienten `k...`, also die Ableitungen dx/dt, für das Runge-Kutta-Verfahren. Die ersten 12 Komponenten von `dxdt` beschreiben die Bewegung der Zentralmassen, also

```
dx12  = xold[0] - xold[6] ;
 ...
rd2 = dx12*dx12 + dy12*dy12 + dz12*dz12 ;
rd  = sqrt(rd2) ;
dxdt[0] = xold[1] ;
dxdt[1] = -1. * m2 * dx12 / rd / rd2 ;
  ...
```

  Für die übrigen `x[]`-Komponenten (Testteilchen) werden jeweils die Abstände zu den beiden Zentralmassen ermittelt und dann analog verfahren.

- Die grafische Ausgabe erfolgt mittels der Funktion `output`. Dabei werden zunächst die zuvor gezeichneten Punkte gelöscht (Vektor `xold[]`) und dann mit den durch `integrate` aktualisierten Positionen `xnew[]` neu gezeichnet. Unter Berücksichtigung des Projektionswinkels `phi`, sind die Projektionen in die Zeichenebene:

```
x =  xnew[n] ;
y =  cos(phi) * xnew[n+2] + sin(phi) * xnew[n+4] ;
```

Untersuchen Sie mittels des fertigen Programms

a) Welchen Einfluss hat der Drehsinn der Galaxien?

b) Was passiert bei unterschiedlichen Massen der Galaxien?

c) Was bewirken verschiedene Inklinationswinkel?

d) Was sieht man unter verschiedenen Projektionswinkeln `phi`?