Fizika IMSC P11 Feladat
Püspök-Kiss Balázs, BL6ADS

Dokumentáció

Feladat: P11. Vízszintes irányú, homogén B mágneses mezőben elengedünk egy Q töltésű, m tömegű pontszerű testet. A testre a nehézségi erőtér (g) is hatással van. Írjunk programot, amely numerikusan kiszámolja és kirajzolja a test pályáját! Mutassuk meg, hogy a pálya ugyanaz, mint egy vízszintes talajon tisztán gördülő henger egy kerületi pontjának pályája (cikloid)!

Megoldás: A testre 2 erő hat, a **nehézségi erő** $F_g = mass * g$ és a **Lorentz erő**, ami ebben az esetben $F_L = Q * B * v$ (ahol B és v vektorok).
Ebből a kettő erőből áll az **eredő erő**, ami $F_e = F_g + F_L$. Newton 2. törvénye szerint
$F_e = mass * acceleration$.
A sebességet a test gyorsulásfüggvényét integrálva kapjuk meg.
A Lorentz erő mindig merőleges a sebességvektorra, irányát a mágneses mező erőssége (B) és a test töltése (Q) adja meg.

Kód: angolul dokumentáltam, mi mit jelent a kódon belül, ez a függvény kezeli a szimulációt

```
void runSim(double b, double q, double m, double time, double frequency) {
	Vector2D MagneticField(b, 0); /// Magnetic Field's Vector in Tesla
	Vector2D Lorentz(0, 0); /// Lorentz Force in Newton
	Vector2D Gravity(0, -9.81 * m); /// Gravitational force (with 9.81 m/s^2 acceleration) in
Newton
	Vector2D CurrentNetForce(0, 0); /// Current Net Force in Newton
	Vector2D CurrentSpeed(0, 0); /// Speed of the Particle in m/s
	Vector2D CurrentAccel(0, 0); /// Acceleration of the Particle in m/s^2
	Vector2D OldAccel(0, 0); /// Previous Acceleration, needed for calculating the speed
	Vector2D CurrentPos(0, 0); /// Starting Position
	std::vector<Vector2D> Pos; /// Positions of the Particle

	Pos.push_back(CurrentPos);

	for (int i = 0; i < time * frequency; i++) {

		/// Calculating the Net Force and Lorentz Force
		/// Lorentz's Force Length is calculated, and then rotated by the Speedvector
		Lorentz = Vector2D(q * b * CurrentSpeed.GetLength(), 0);
		Lorentz.RotateDeg(CurrentSpeed.GetRotationDeg() - 270);
		CurrentNetForce = Gravity + Lorentz; /// A simple vector addition

		/// Calculating Velocity and Acceleration
		double two = 2;
		OldAccel = CurrentAccel;
		CurrentAccel = CurrentNetForce / m;
		CurrentSpeed += (OldAccel+CurrentAccel) / frequency / two;

		/// Calculating Position
		CurrentPos += CurrentSpeed / frequency;
		Pos.push_back(CurrentPos);
	}
	SaveSim(Pos); /// Saves the Simulation to "Simulation.dat"
	functionDraw(Pos); /// Draws the Simulation to a Window
}
```

Fizika IMSC P11 Feladat
Püspök-Kiss Balázs, BL6ADS

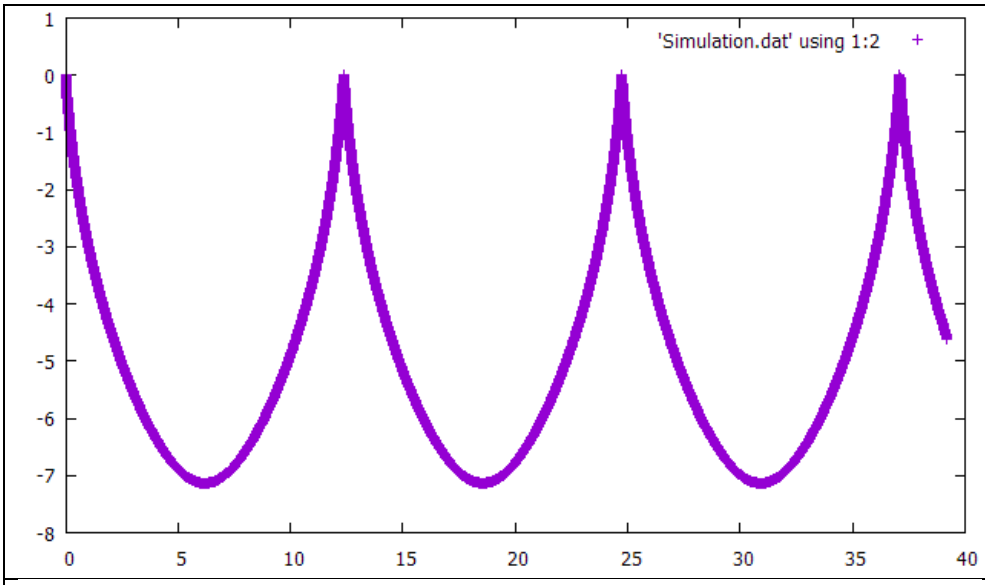
A Vector2D egy sajátkészítésű osztály, ami vektorokat tárol és megadja az adott vektor tulajdonságait.

```
class Vector2D {
	double x;
	double y;
public:
	Vector2D(double x = 0, double y = 0) : x(x), y(y) { } /// Default Constructor
	~Vector2D() { } /// Default Destructor

	/// Getters, Setters
	void SetX(double nX) { x = nX; }
	void SetY(double nY) { x = nY; }
	double GetX() { return x; }
	double GetY() { return y; }


	double GetLength() {
		if (x == 0 && y == 0)
			return 0;
		return sqrt(exp2(abs(x)) + exp2(abs(y)));
	}
	double GetRotationDeg() { /// Gets the rotation of the vector, gives back a Degree
		if (y == 0) { /// Checking if tangent exists
			if (x >= 0)
				return 90;
			else
				return -90;
		}
		if ((atan2(y, x)) / M_PI * 180 >= 0) /// Checking cases
			return (atan2(y, x)) / M_PI * 180;
		return 360 + (atan2(y, x)) / M_PI * 180;
	}
	void RotateDeg(double deg) { /// Rotates Vector by Degree
		double NewX = x * cos(deg / 180 * M_PI) - y * sin(deg / 180 * M_PI);
		double NewY = x * sin(deg / 180 * M_PI) + y * cos(deg / 180 * M_PI);
		x = NewX;
		y = NewY;
	}

	///Operator things
	Vector2D operator+(Vector2D& rhs) { return Vector2D(rhs.GetX() + x, rhs.GetY() + y); }
	Vector2D operator/(double& rhs) { return Vector2D(x / rhs, y / rhs); }
	Vector2D operator*(double& rhs) { return Vector2D(x * rhs, y * rhs); }
	void operator+=(Vector2D rhs) {
		x += rhs.GetX();
		y += rhs.GetY();
	}
};
```

Egyébként csináltam egy pontábrázolót is, ami nem tökéletes, de mikor a program leszimulálta az esetet, utána megjeleníti az ablakban rácsokkal az eredményt. GNUPlottert fogok használni a az adataim leteszteléséhez.

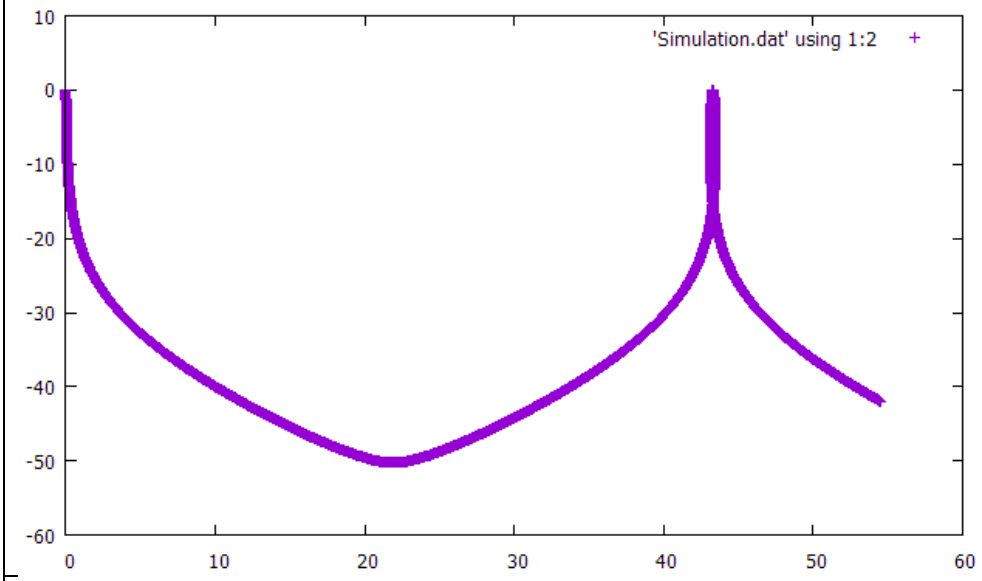
Fizika IMSC P11 Feladat
Püspök-Kiss Balázs, BL6ADS

Szimulációk adott paraméterekkel



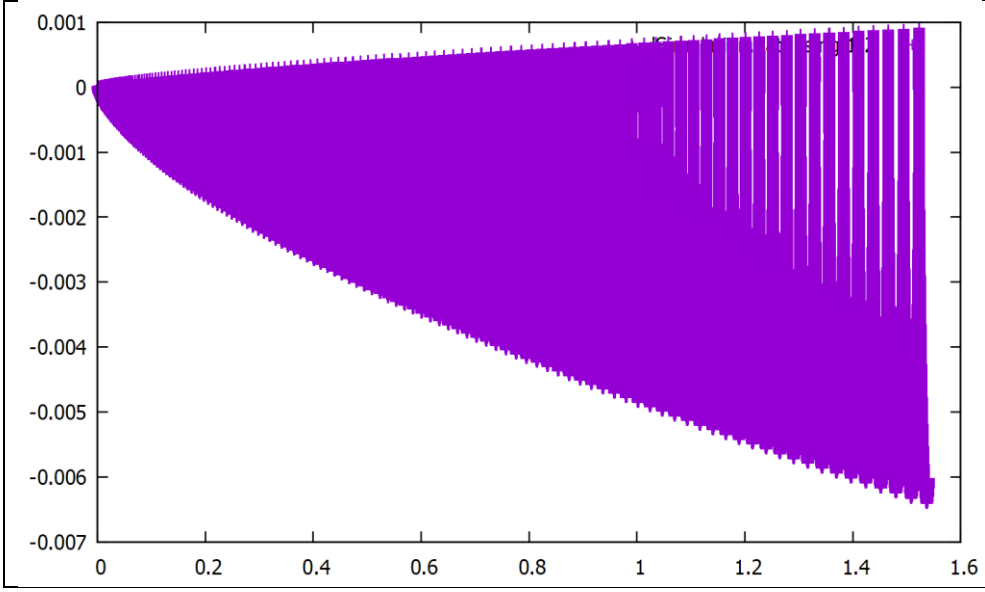
B = 1T
Q = 1C
m = 1kg
g = 9.81m/s^2

Szépen kivehető a cycloid minta, amit írt a feladat.



B = 0.01T
Q = 0.01C
m = 0.01kg
g = 9.81m/s^2

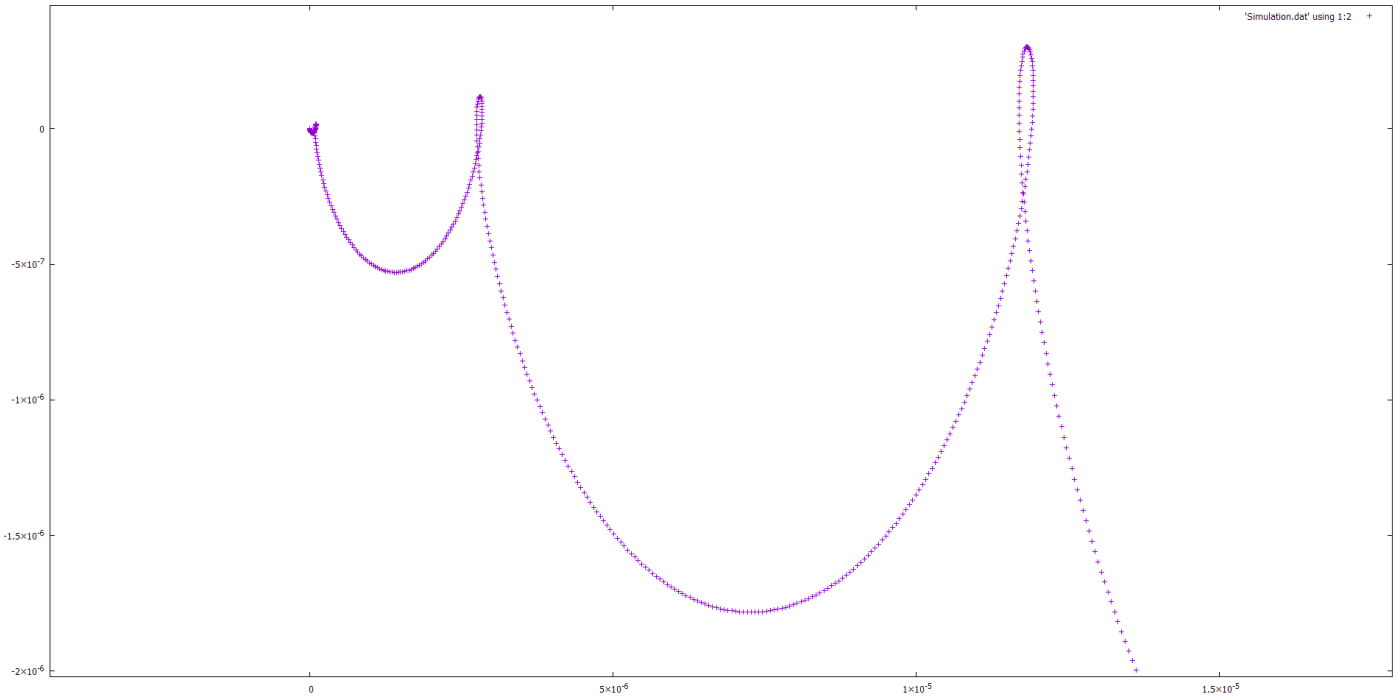
Mi történik akkor, ha a gravítáció kerül fölénybe?



B = 100T
Q = 1000C
m = 1000kg
g = 9.81m/s^2

Esetleg mi történik akkor, ha thor horgonyát elengedjük az űrben egy feketlyuk felé száguldva (9.81m/s^2 gyorsulással)

Fizika IMSC P11 Feladat
Püspök-Kiss Balázs, BL6ADS



Érdekes, hogy ha nagyon közel nagyítunk a függvény (B=0.1T, Q=10C, m=0.1kg) röppályájára, akkor mégsem teljesen a cycloid függvényt kapjuk eredményül.