گزارش کار تمرین نهم شبیه‌سازی رایانه‌ای در فیزیک

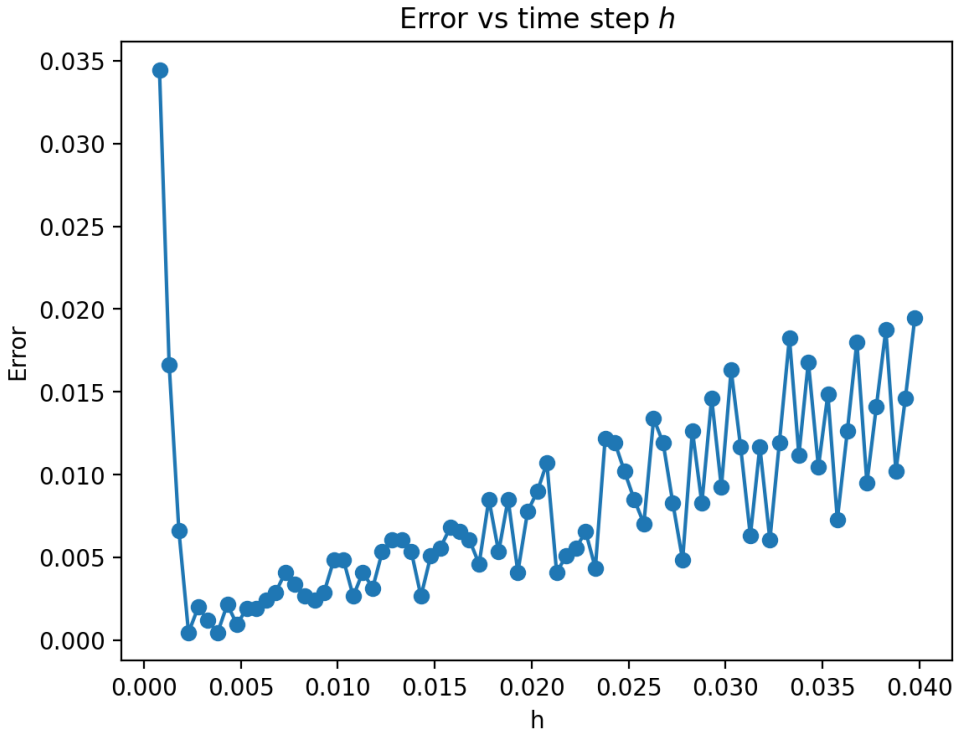
علی اکرامیان – 99100563

بخش اول)

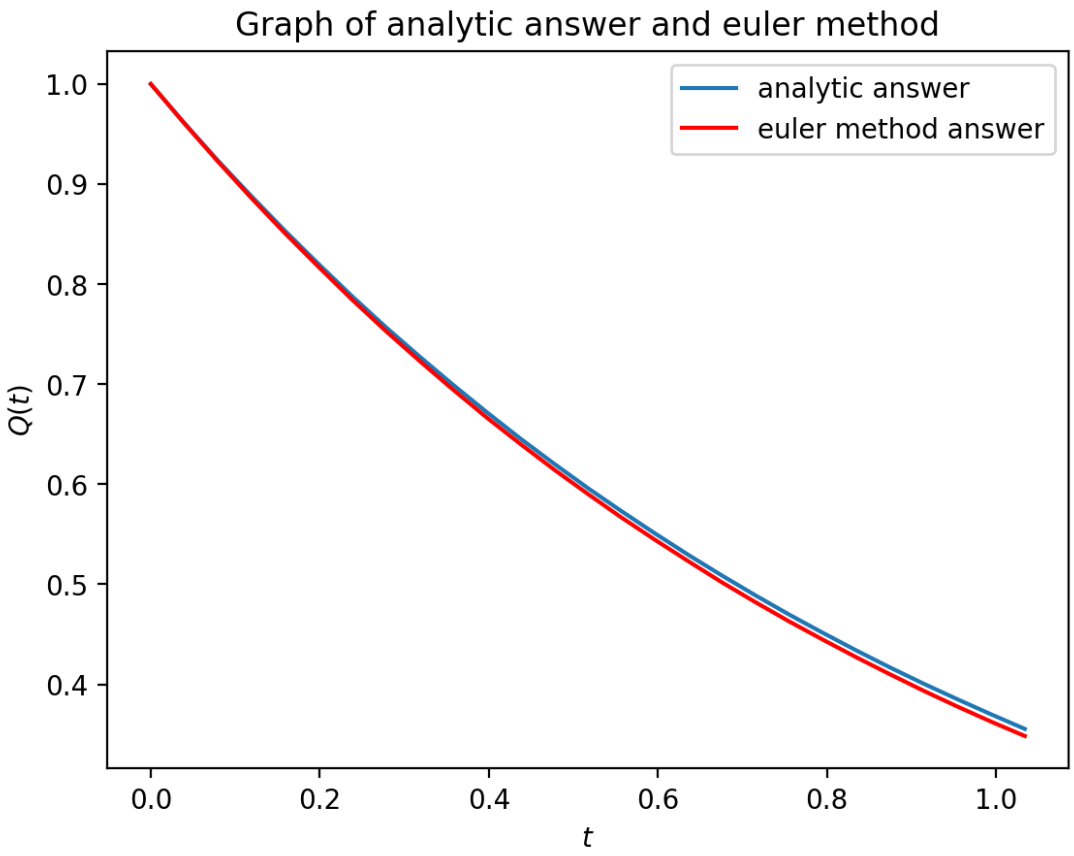
در این تمرین من ابتدا معادله را نوشتم و R و C را یک قرار دادم. سپس زمانی که تا آن زمان حل می‌کنم و مقایسه را در آن زمان انجام می‌دهم، T می‌گیرم. حال یک تابع f(Q,t) = dQ/dt تعریف می‌کنم که مشتق کیو (بار) را بازگرداند (return) سپس یک تابع Q(t) تعریف می‌کنم که جواب تحلیلی است که مقایسه را با آن انجام می‌دهم. حال یک آرایه از h ها تعریف می‌کنم که رنج آن را از تجربه آوردم (چندین بازه‌ی مختلف را تست کردم و در این بازه‌ای که گذاشتم بنظرم بهترین شکلی بود که می‌شد داشت و سر کلاس نیز به این شکل اشاره شده بود. یک ارایه‌ی error هم تعریف می‌کنم که همه‌ی درایه‌هایش صفر است. در ضمن همه‌ی این‌ها را float16 گرفتم تا در اچ‌های بزرگ بتون نمودار را دید. اگر از دیگر دیتا تایپ‌ها استفاده کنیم، باید اچ‌ها را خیلی کوچک‌تر کنیم تا این نمودار را ببینیم. حال یک حلقه می‌زنم روی تمام اچ‌ها و یک آرایه‌ی زمانی داریم که اچ‌تا اچ‌تا از 0 تا T پیش می‌رود که زمان نهایی من است. یک آرایه‌ی q هم می‌گذارم. همه‌ی این‌ها نیز float16 هستند. حال شرط اولیه‌ی q[0] = Q0 را می‌گذارم و الگوریتم اویلر را اجرا می‌کنم. به این ترتیب که هر مرتبه، بار جدید q[j] برابر می‌شود با بار قبلی q[j-1] به علاوه‌ی اچ ضرب در شیب تغییر بار f(Q,t). پس از اتمام الگوریتم به ازای آن اچ خاص، خطا را توسط قدرمطلق اختلاف بار در زمان T یعنی q[T] که همان درایه‌ی آخر این آرایه است q[-1] با مقدار واقعی که در تابع Q(t) قرار دارد.

بعد از محاسبات، این نمودار خطا را برحسب اچ‌ها می‌کشم. یک نمودار دیگر هم می‌کشم که نمودار Q(t) است به همراه چیزی که با اویلر حساب کرده بودیم یعنی نمودار q.

شکل‌ها را می‌بینیم:

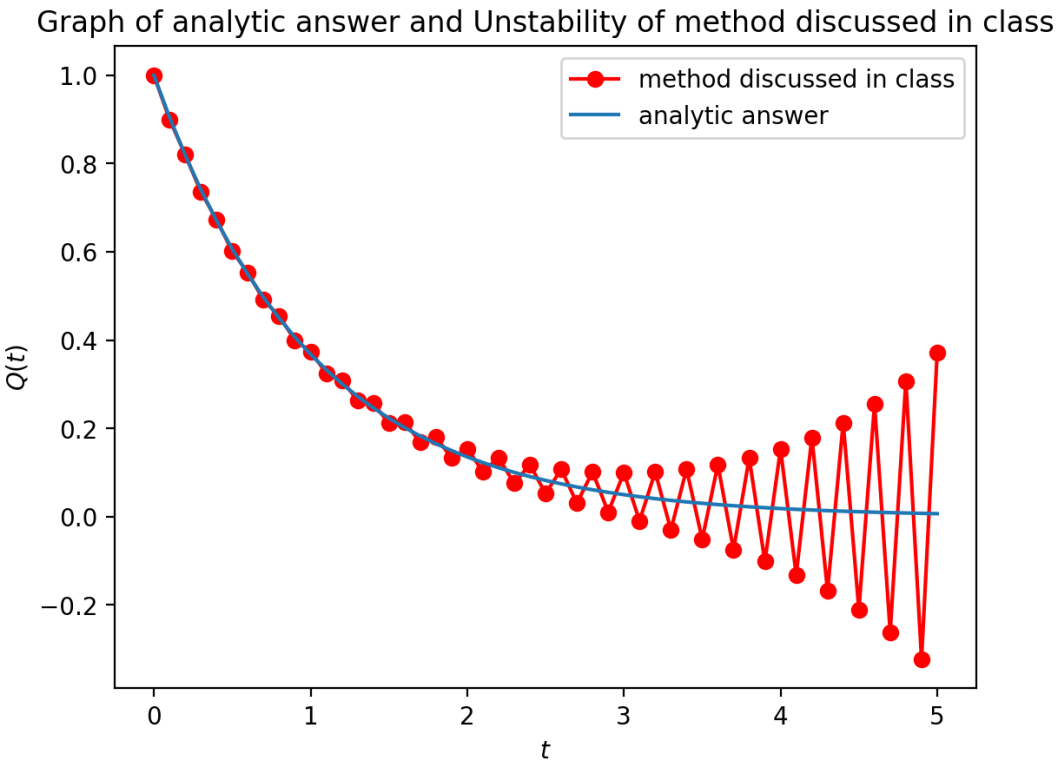


و شکل تابع نهایی به همراه جواب تحلیلی:



بخش دوم)

در این تمرین من همان مقادیر اولیه را در نظر گرفتم و توابع بخش قبل نیز همان‌ها هستند. حال اچ را مثلا 0.1 می‌گیرم (که طبعا در کد قابل تغییر است!) و مانند قبل یک آرایه‌ی زمان‌ها t و بارها q تعریف می‌کنم و شرط اولیه q[0] = Q0 را قرار می‌دهم. حال چون قرار است این بار این معادله را با الگوریتم y(n+1) = y(n-1)+2f(n)h حل کنیم، یک قدم اویلری اول برمیدارم (q[1]=q[0]+h*f(q[0],t[0]) سپس حلقه‌ای میزنم و الگوریتم پیشنهادی را اجرا می‌کنم. حال چون این الگوریتم مطابق بحث‌های سر کلاس یک جمله‌ی رشد کننده‌ی نوسانی دارد مرتب این جمله + و − شده و این الگوریتم ناپایدار است. سپس شکل را نیز رسم می‌کنم. که در شکل حل این معادله به روش الگوریتم ناپایدار بالا و حل دقیق تحلیلی را با هم رسم کرده‌ام.
شکل را در صفحه‌ی بعد می‌بینیم.



بخش سوم)

حال به نوسان‌گر هماهنگ ساده می‌پردازیم با چندین الگوریتم مختلف! ابتدا من یک w تعریف کرده‌ام که همان فرکانس زاویه‌ای است. سپس زمان نهایی که حل تا آن زمان انجام می‌شود را برابر T گذاشته‌ام. سپس یک آرایه‌ی Hs تعریف کردم که با چند اچ مختلف بتوان کد را اجرا کرد و تفاوت الگوریتم‌ها را در اچ‌های متفاوت مشاهده کرد.
حال این معادله‌ی مرتبه 2 را به شکل یک دستگاه مرتبه 1 می‌نویسم.

$$x'' = -w^2 x \Rightarrow \begin{cases} x' = X(x,v,t) = v \\ v' = V(x,v,t) = -w^2 x \end{cases}$$

حال دو تابع X(x,v,t) و V(x,v,t) را در کد تعریف می‌کنم که مشتقات سرعت و مکان هستند. حال تابعی دیگر که تابع انرژی است تعریف می‌کنم که انرژی جنبشی به علاوه‌ی پتانسیل است و به شکل زیر است:

$$E = \frac{1}{2}mv^2 + \frac{1}{2}kx^2$$

حال دو تابع ans_x و ans_v نیز تعریف می‌کنم که جواب‌های تحلیلی نوسانگر هماهنگ هستند (جواب‌های دقیق)
در این کد من شرایط اولیه‌را نیز x0=1 و v0=1 قرار دادم و w=1 قرار دادم که طبعا در کد قابل تغییر اند.

حال یک حلقه روی اچ‌های مختلف می‌زنم تا به ازای چند اچ خروجی‌ها را داشته باشیم. در این حلقه نیز حل اصلی معادله با 5 الگوریتم وجود دارد. ابتدا یک آرایه‌ی زمان t تعریف می‌کنم که مانند بخش‌های قبل است. سپس یک ارایه‌ی x و یک آرایه‌ی v برای هر کدام از الگوریتم‌ها تعریف می‌کنم که از اسم‌هایشان مشخص است و مکان‌ها و سرعت‌های بدست آمده از هر الگوریتم در آرایه‌ی مربوط به خودشان ریخته می‌شود. جواب‌های الگوریتم اویلر در x_euler و v_euler و الگوریتم اویلر-کرامر در x_ek و v_ek و الگوریتم ورله در x_verle و v_verle و الگوریتم بیمن در x_beeman و v_beeman و همچنین پرش قورباغه‌ای نیز در x_frog و v_frog ریخته می‌شود. حال شروط اولیه را باید قرار دهیم. درایه‌های اول مکان‌ها را x0 و سرعت‌ها را v0 می‌گذاریم (در تمامی الگوریتم‌ها). سپس برای الگوریتم‌های بیمن و پرش غورباقه‌ای چون به قدم‌های 1 نیز نیاز داریم قدم اول‌شان را با اویلر برمیداریم و سپس حلقه‌ی اصلی را روی زمان می‌زنیم.
در این حلقه که حل اصلی در این حلقه انجام می‌شود، الگوریتم‌های حل هر 5 الگوریتم را نوشته‌ام. اندیس‌ها را نیز طوری قرار دادم که هر 5 تا را بتوانم در یک حلقه حل کنم و کد تمیز و بهره‌ور بشود!
الگوریتم‌ها را مطابق همان‌ها که در کلاس بود نوشتم و اینجا صرفا همان‌ها را می‌اورم:

الگوریتم اویلر:

```
x_euler[j]=x_euler[j-1]+h*X(x_euler[j-1],v_euler[j-1],t[j-1])
v_euler[j]=v_euler[j-1]+h*V(x_euler[j-1],v_euler[j-1],t[j-1])
```

الگوریتم اویلر-کرامر:

```
v_ek[j]=v_ek[j-1]+h*V(x_ek[j-1],v_ek[j-1],t[j-1])
x_ek[j]=x_ek[j-1]+h*X(x_ek[j-1],v_ek[j],t[j])
```

الگوریتم ورله:

```
x_verle[j]=x_verle[j-1]+h*X(x_verle[j-1],v_verle[j-1],t[j-1])+0.5*V(x_verle[j-1],v_verle[j-1],t[j-1])*h*h
v_verle[j]=v_verle[j-1]+0.5*h*(V(x_verle[j-1],v_verle[j-1],t[j-1])+V(x_verle[j],v_verle[j],t[j]))
```

الگوریتم بیمن:

```
x_beeman[j+1]=x_beeman[j]+h*X(x_beeman[j],v_beeman[j],t[j])
+h*h*(4*V(x_beeman[j],v_beeman[j],t[j])-V(x_beeman[j-1],v_beeman[j-1],t[j-1]))/6
v_beeman[j+1]=v_beeman[j]+h*(2*V(x_beeman[j+1],v_beeman[j+1],t[j])-V(x_beeman[j-1],v_beeman[j-1],t[j-1])+5*V(x_beeman[j],v_beeman[j],t[j]))/6
```

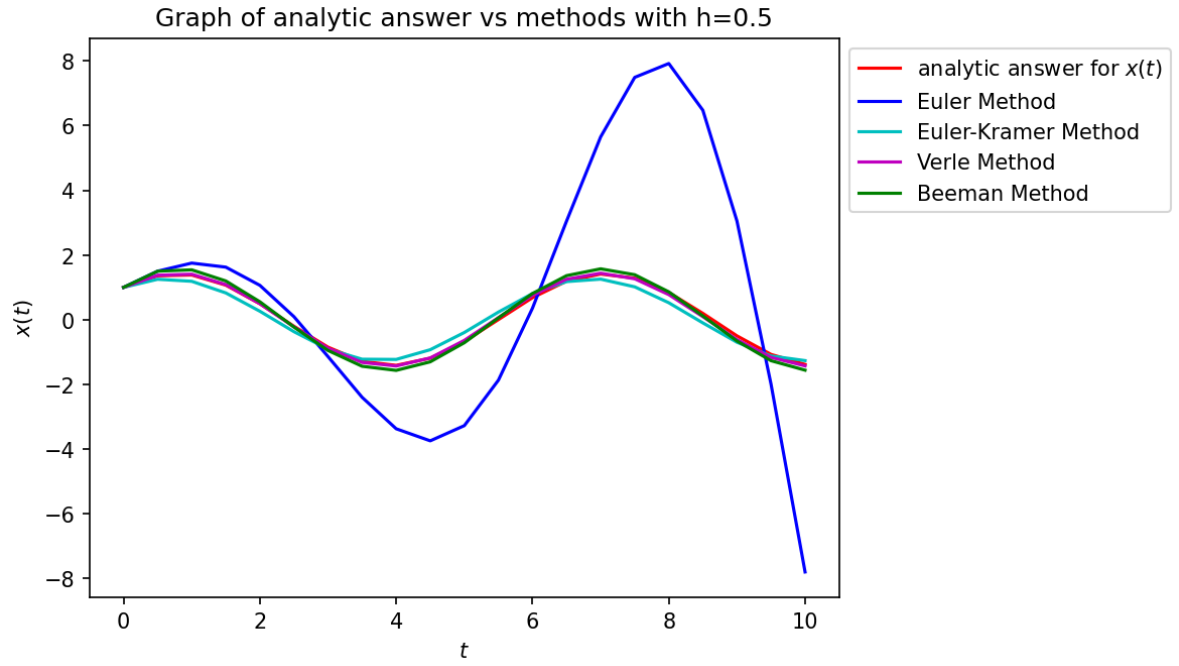
الگوریتم پرش قورباغه‌ای:

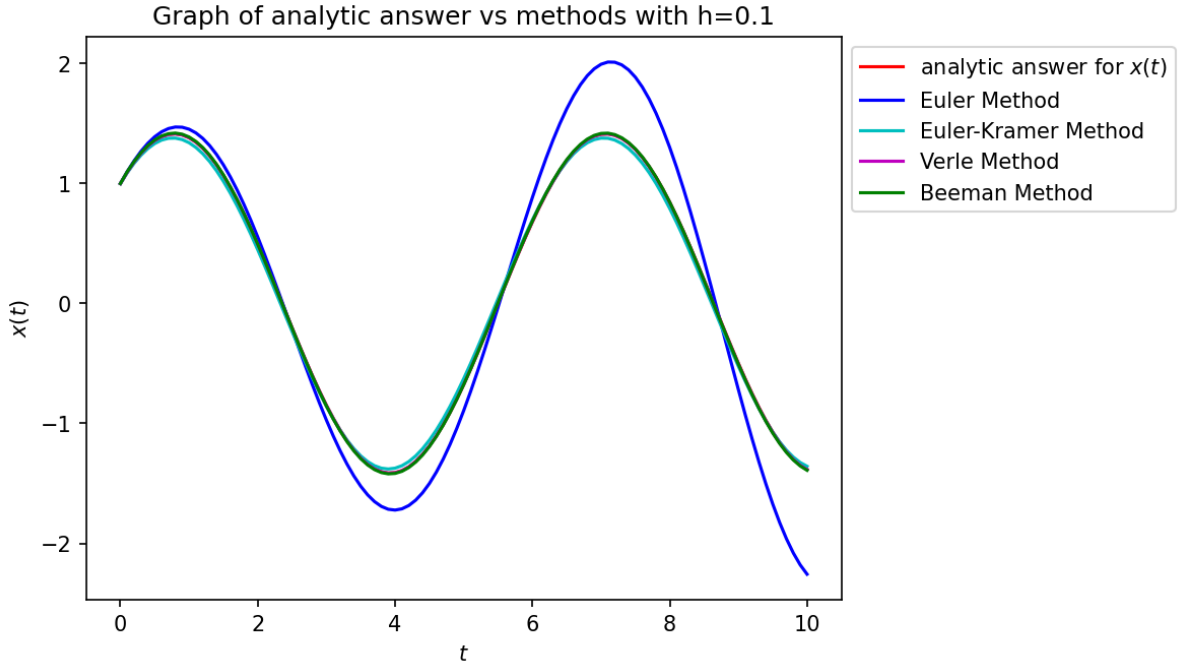
```
x_frog[j+1]=x_frog[j-1]+X(x_frog[j],v_frog[j],t[j])*h
v_frog[j+1]=v_frog[j-1]+V(x_frog[j],v_frog[j],t[j])*h
```

که در الگوریتم پرش قورباغه‌ای چون سر کلاس با عددهای غیر صحیح گفته شده بود من اعداد اندریس را صحیح کردم ولی برای گرفتن جواب تا زمان T باید تا زمان 2T برنامه را اجرا کرد. در رسم‌ها به دلیل شباهت‌های این الگوریتم به اویلر-کرامر و نزدیکی به بقیه من این الگوریتم را حل‌هایش را نشان ندادم.

حال به سراغ رسم می‌رویم.

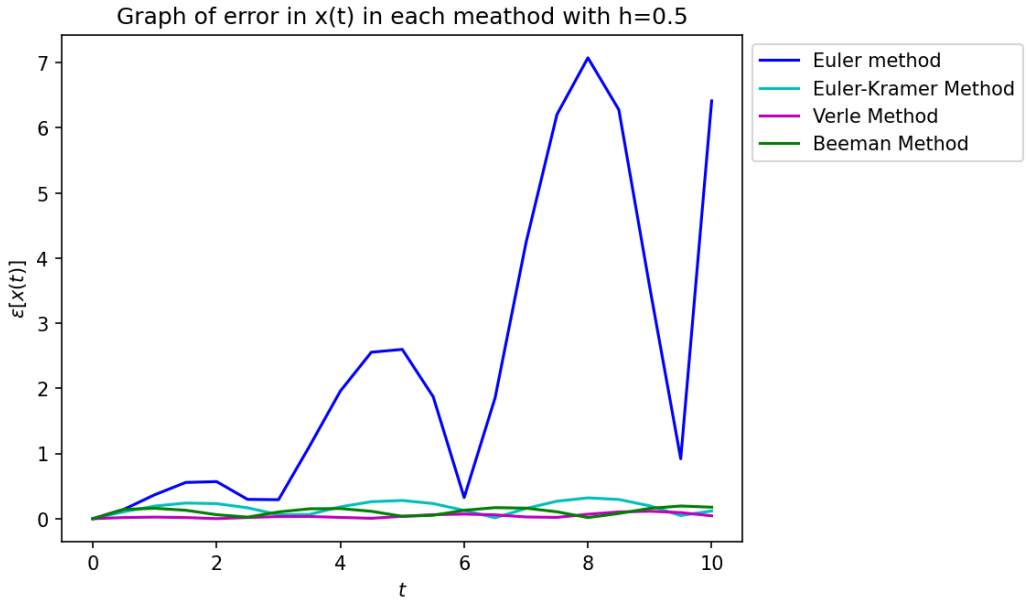
ابتدا در پرش قورباغه‌ای و بیمن گام‌های دوتا به جلو داریم یک دانه بیشتر رفته است که من با دستور اخرین درایه‌ی دو ارایه‌ی مربوط به مکان‌های پرش و بیمن را جذف می‌کنم تا زمان و مکان طول آرایه‌ی برابر داشته باشند.
حال اولین نموداری که رسم می‌کنم نمودار مکان بر حسب زمان x(t) است. که هر رنگ نشان دهنده‌ی یک الگوریتم است که در شکل لیبل نیز خورده اند. در ضمن جواب تحلیلی را نیز با رنگ قرمز رسم کرده‌ام.
شکل را می‌بینیم:

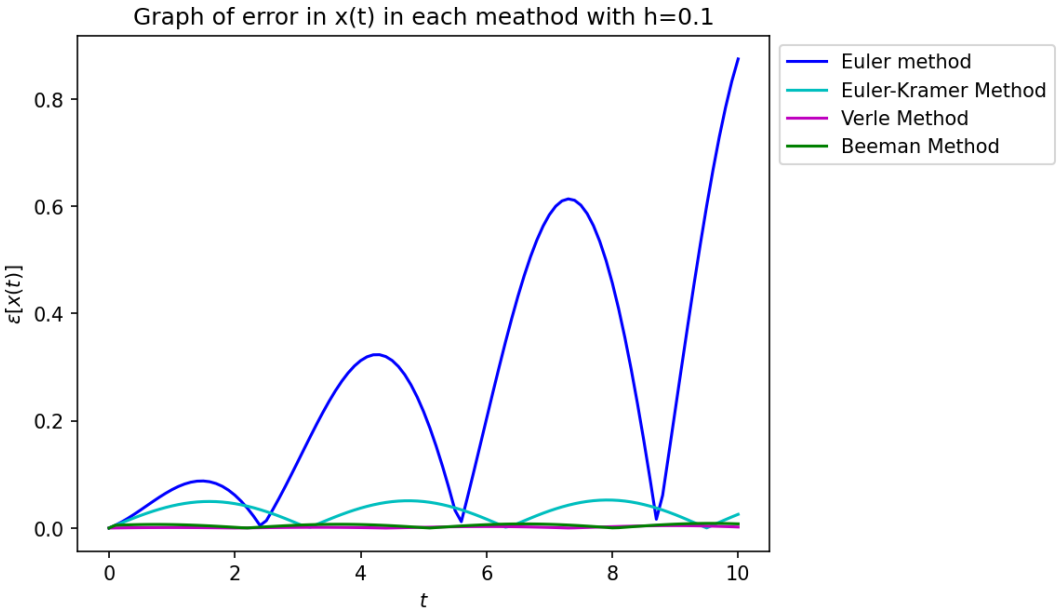




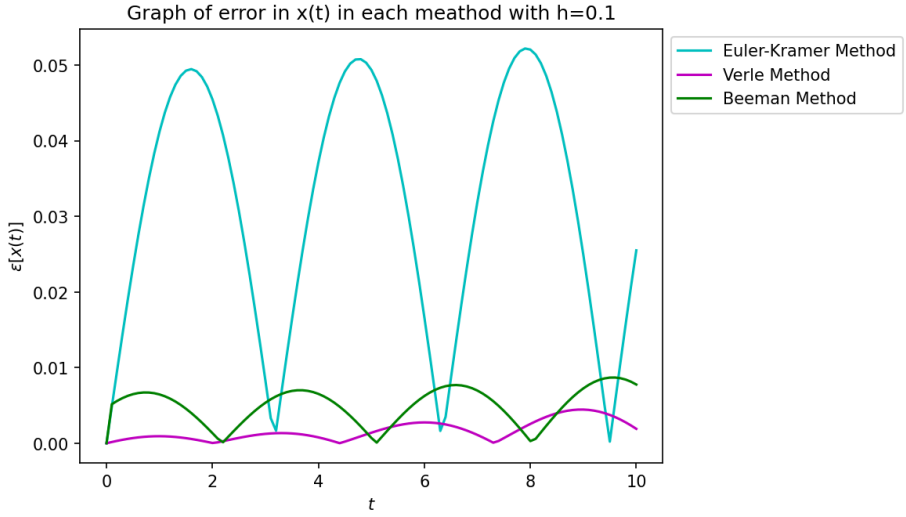
در کد برای h=0.01 نیز رسم شده است.

دومین نموداری که رسم کردم نمودار قدر مطلق خطا (انحراف از مقدار واقعی) بر حسب زمان است.

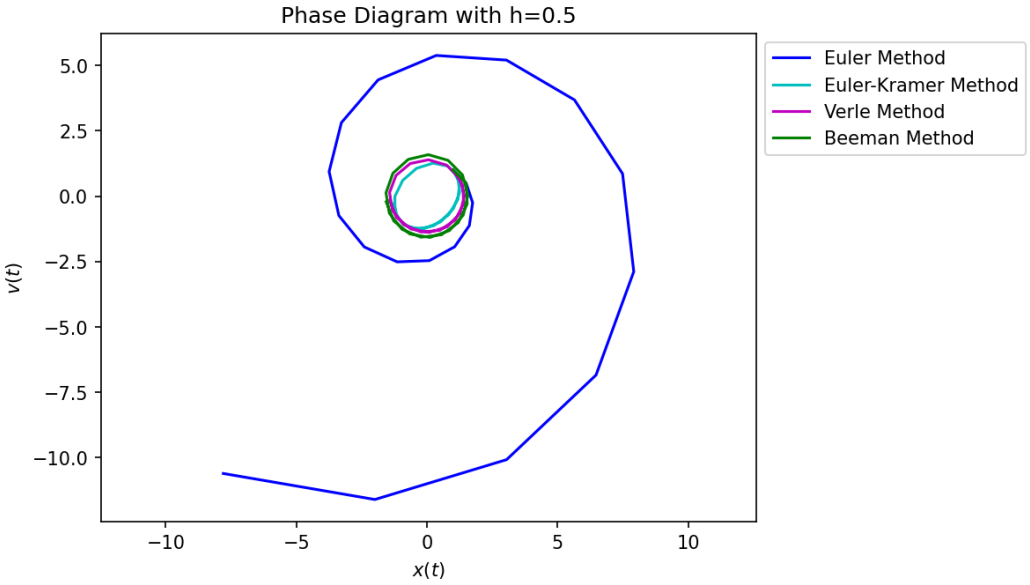


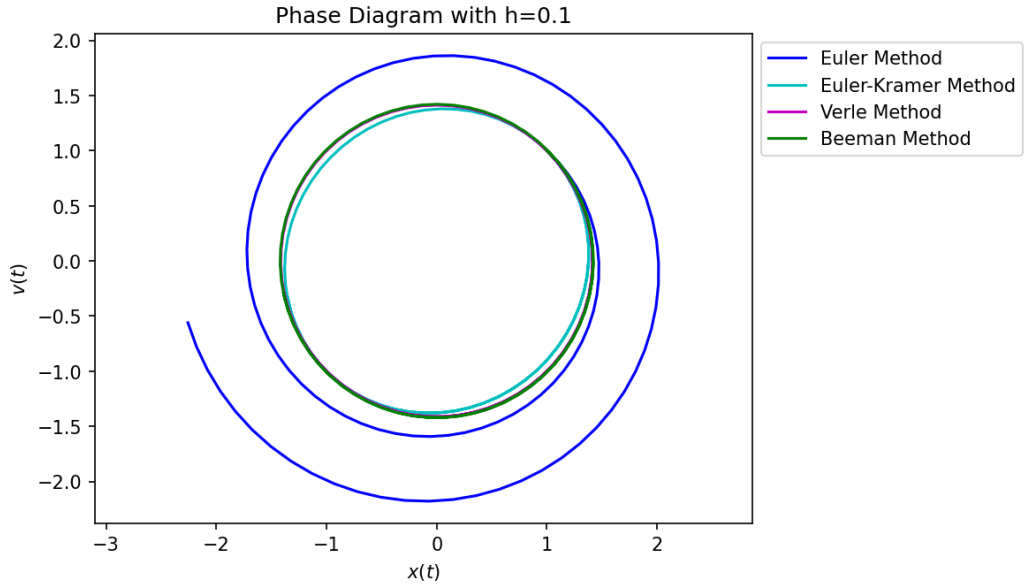


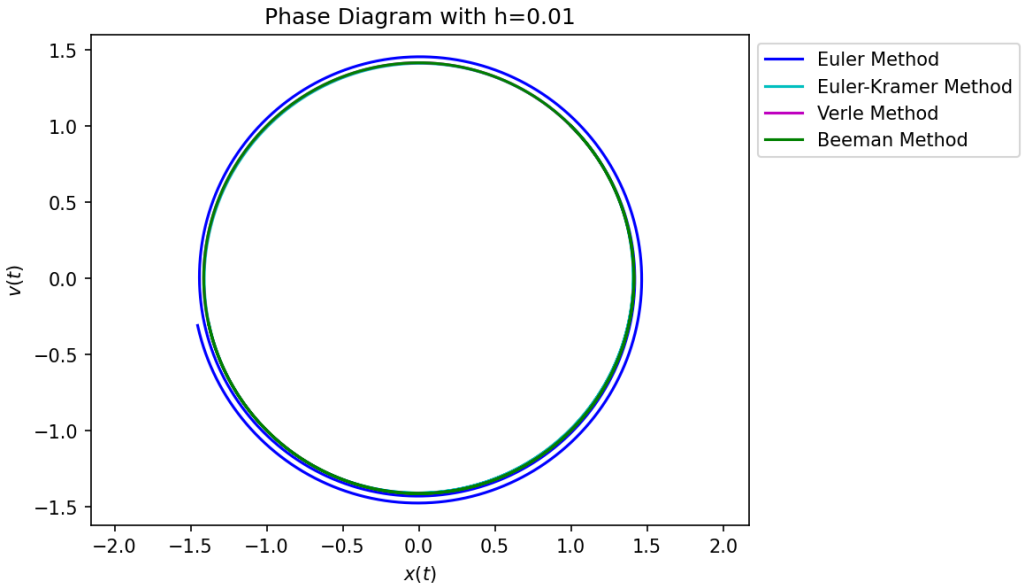
حال می‌توان اویلر را به دلیل اختلاف زیاد خاموش کرد و بدون آن رسم کرد: (برای h=0.1)



سومین نموداری که کشیدم نمودار فضای فاز است که برای الگوریتم‌های مختلف آمده است:

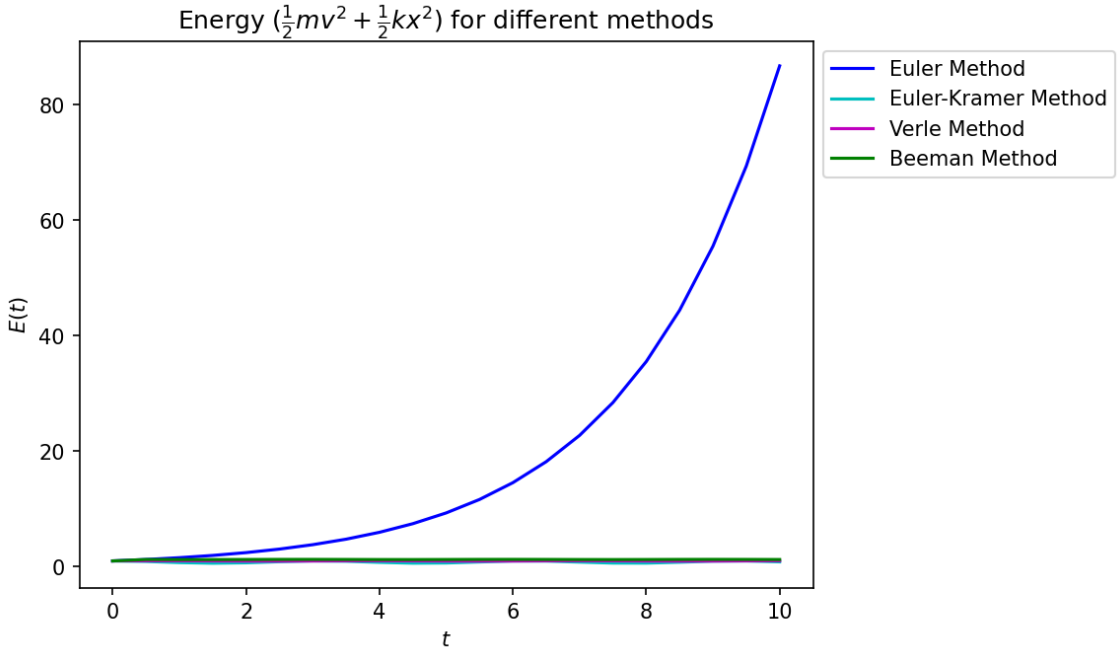




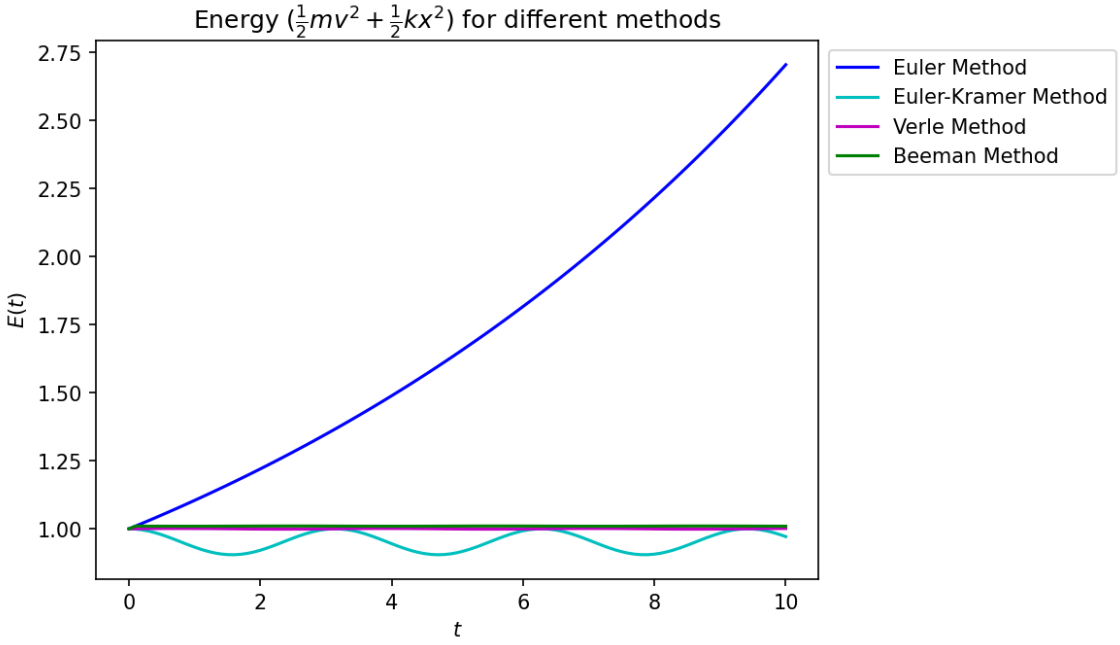


که علی الصول باید یک دایره (یا بیضی) ثابت باشد ولی چون الگوریتم‌ها جواب دقیق نمی‌دند می‌بینیم که مثلا اویلر چرخشی است و دارد بزرگ می‌شود. چون انرژی را ثابت نگه نمی‌دارد و انرژی آن مرتبا دارد زیاد می‌شود.

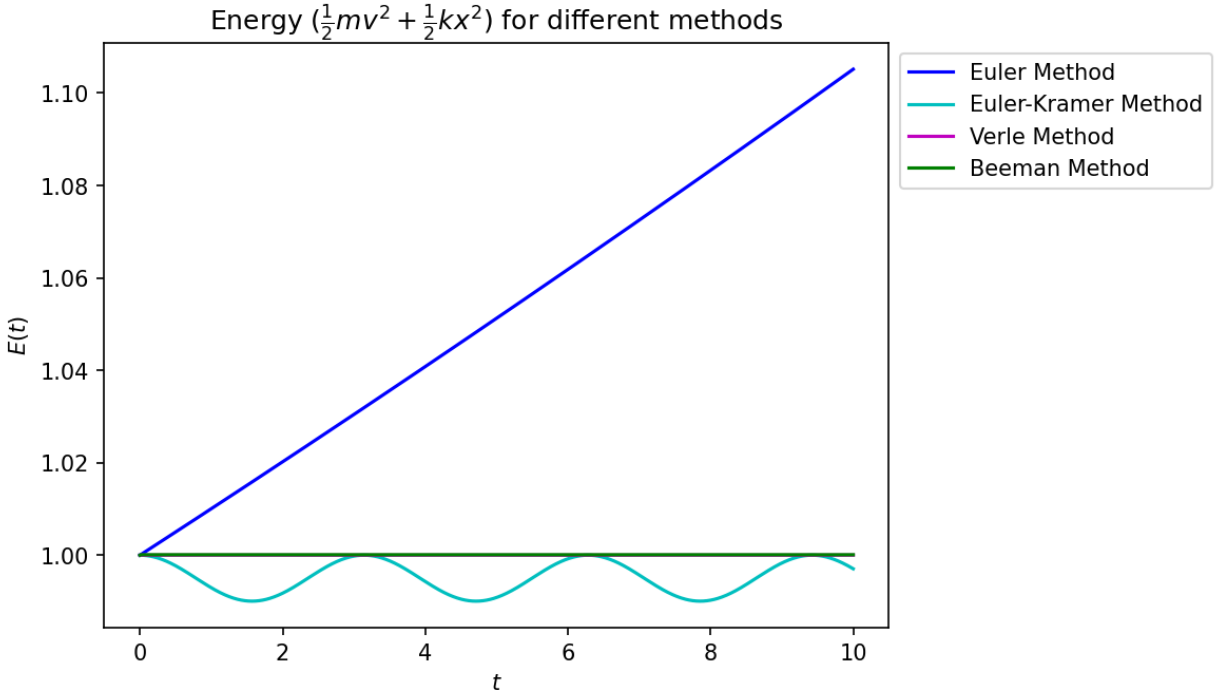
چهارمین نموداری که رسم کردم نیز نمودار انرژی بر حسب زمان است که انتظار داریم پایسته باشد ولی چون الگوریتم‌ها خطا دارند نسبت به حل دقیق پس انرژی نیز کم یا زیاد می‌شود. یا مثل اویلر-کرامر نوسان می‌کند.
شکل‌ها را می‌بینیم:
برای h=0.5



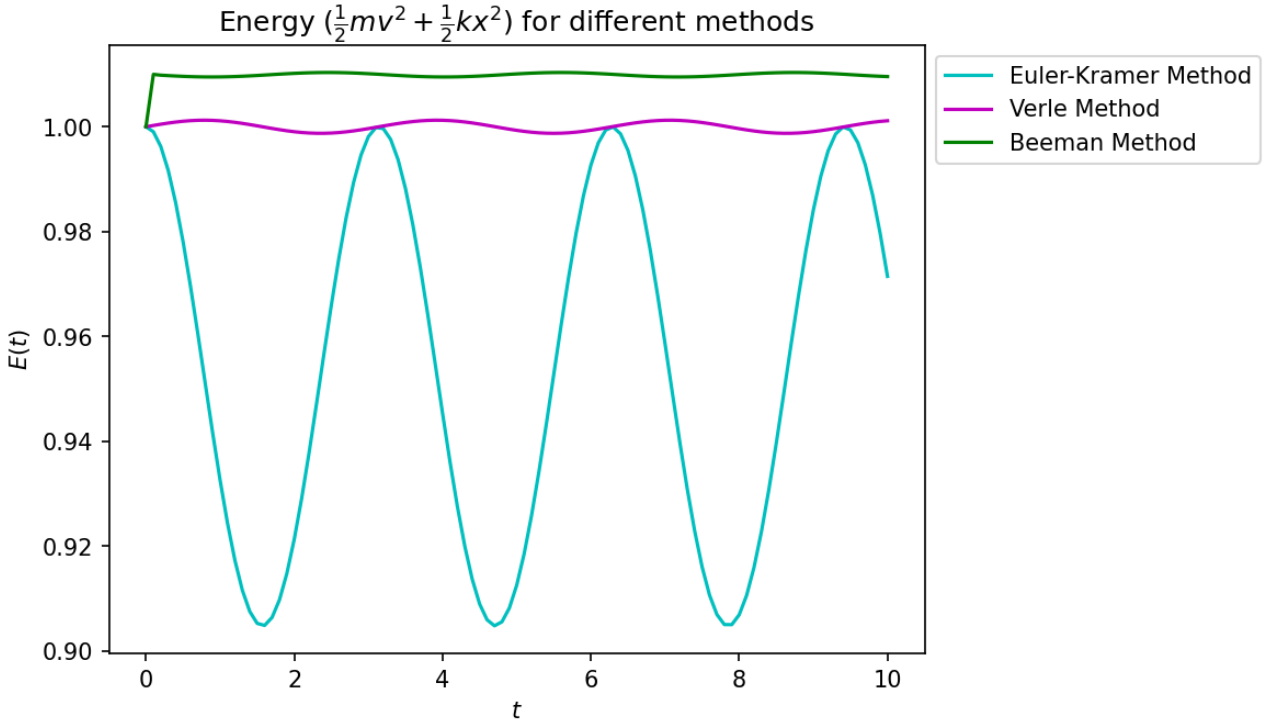
برای h=0.1



برای h=0.01



و برای این‌که تفاوت بین بقیه‌ی الگوریتم‌ها واضح‌تر شود اویلر که خیلی پایستگی انرژی را به هم میریزد خاموش می‌کنم: (برای h=0.1)



که مشخص است الگوریتم بیمن نوسان انرژی کمتری دارد و بهتر است. این که مقداری بالاتر از 1 رفته نیز به خاطر این است که گام اول بیمن را با اویلر برداشته ایم. و از وقتی که بیمن اجرا می‌شود خیلی بهتر از اویلر و بقیه است.