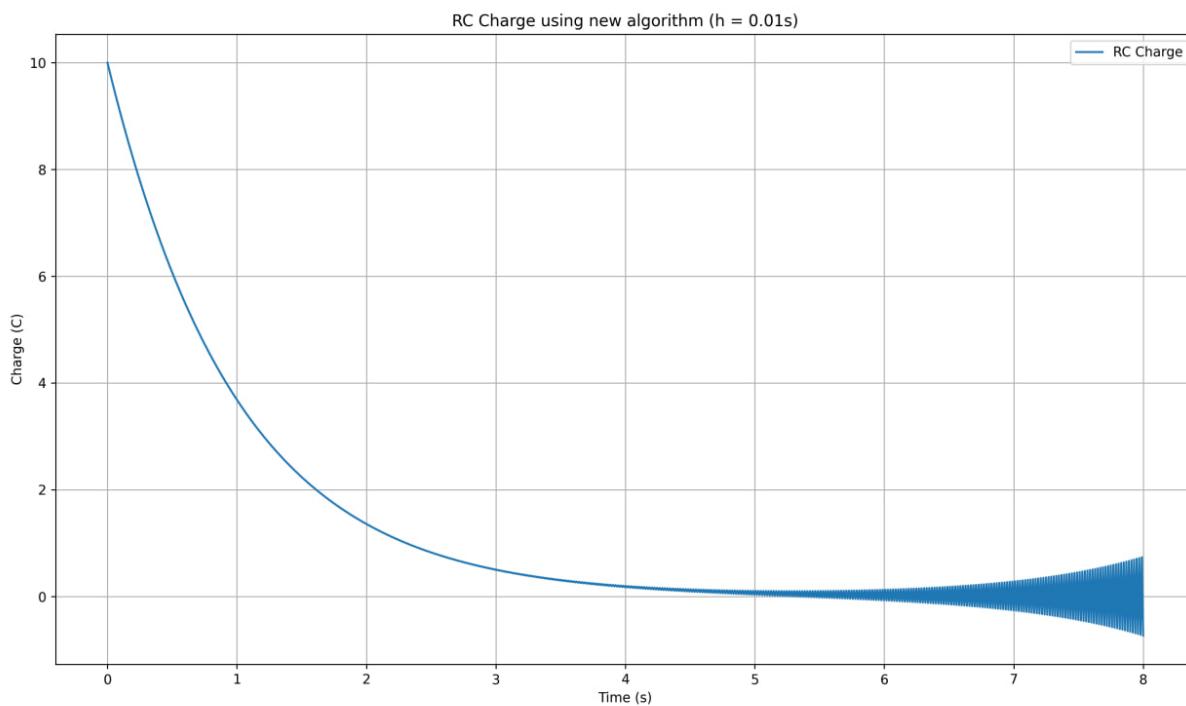


# Algorithms & Chaos

## 1- ناپایداری الگوریتم: alg\_instability.py

تمرین خازن سری پیش را با الگوریتم جدید می نویسیم. نمودار دی شارژ شدن خازن ( $V=0$ ) را رسم می کنیم.

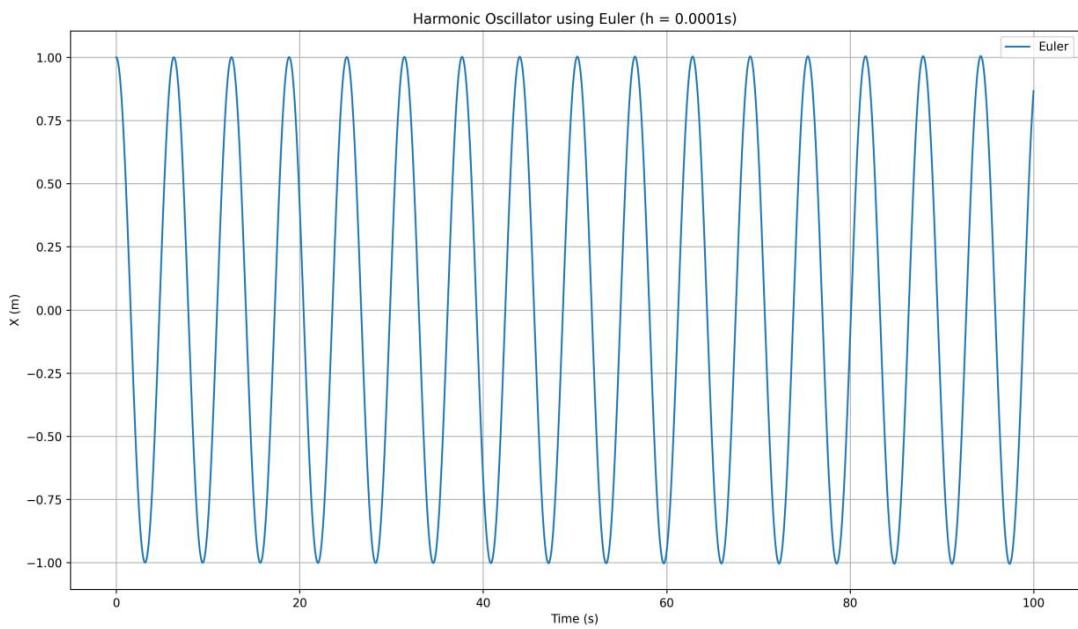


طبق انتظار مشاهده می کنیم که این الگوریتم برای زمان های زیاد، ناپایدار و آشوبناک است.

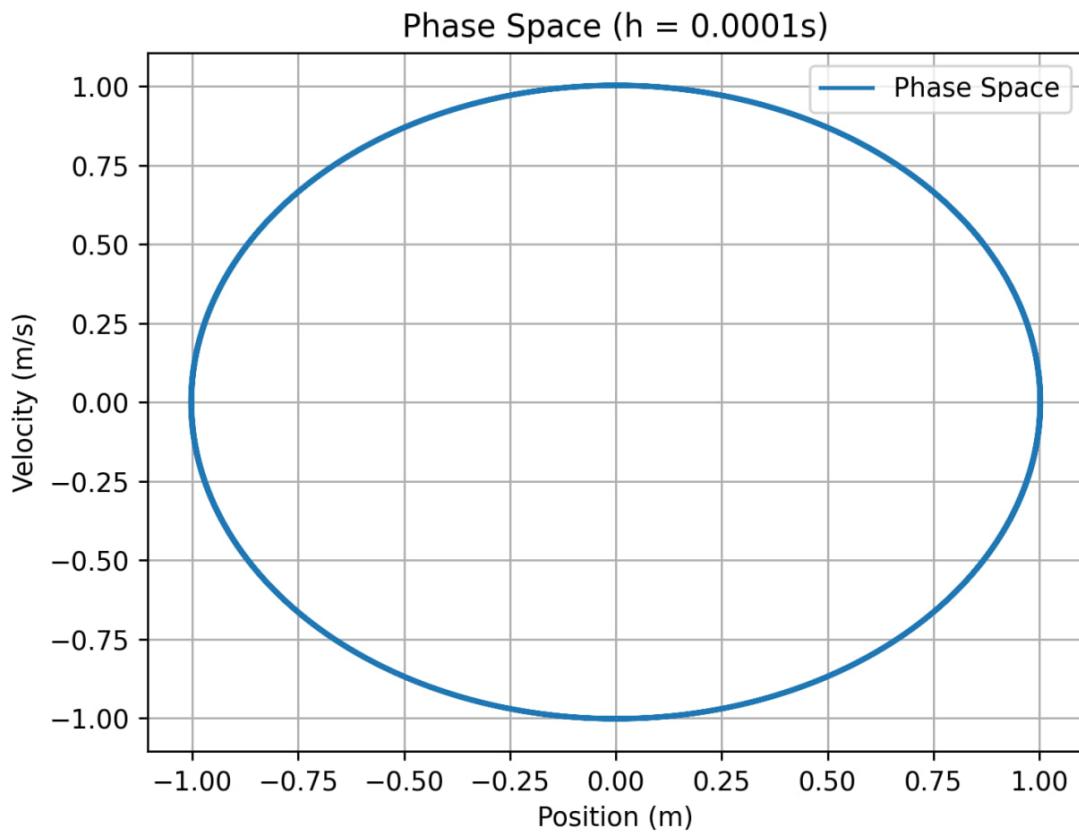
## 2- مقایسه الگوریتم ها: oscillator.py

معادله نوسانگر را با الگوریتم های خواسته شده پیاده سازی می کنیم و  $V$ ,  $X$  را در آرایه هایی ذخیره می کنیم. سپس نمودار مکان برحسب زمان و نمودار فضای فاز را رسم می کنیم.

اویلر:

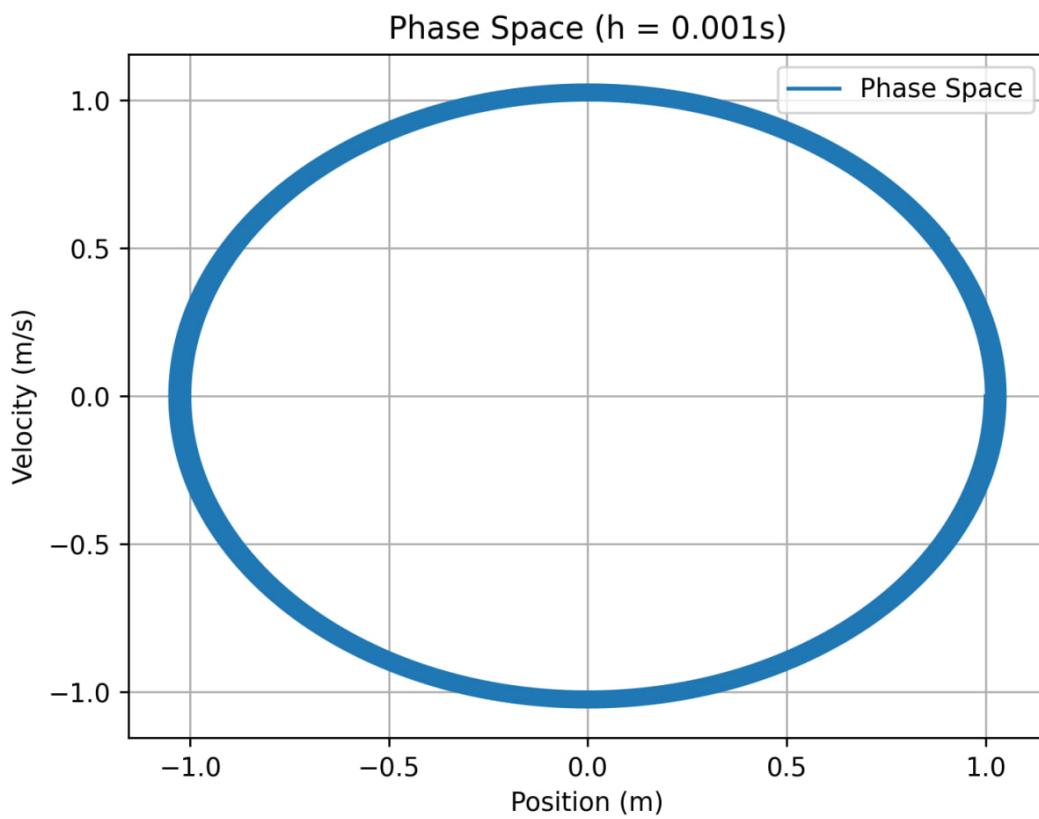
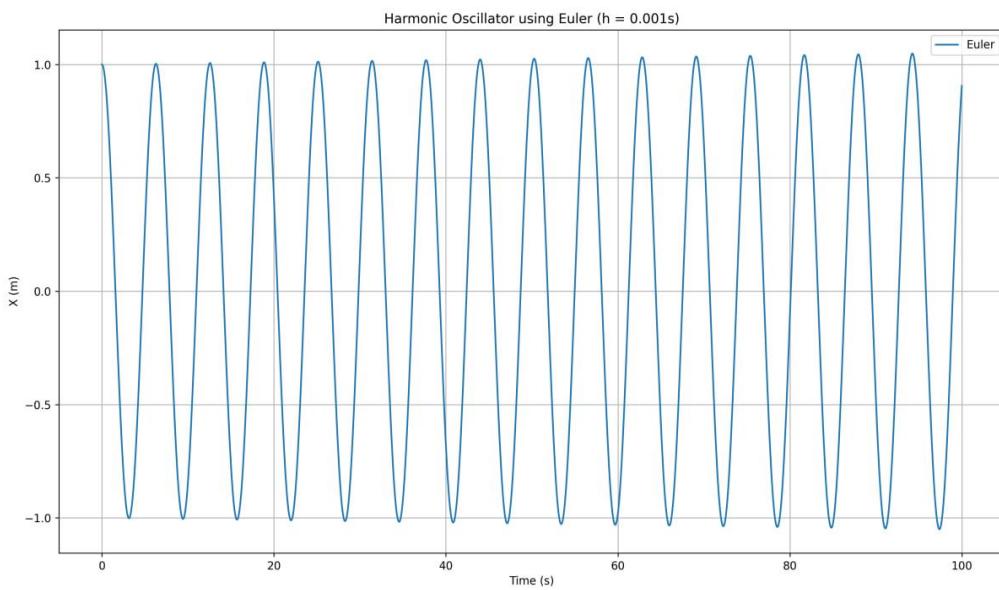


رفتار نوسانی قابل مشاهده است.

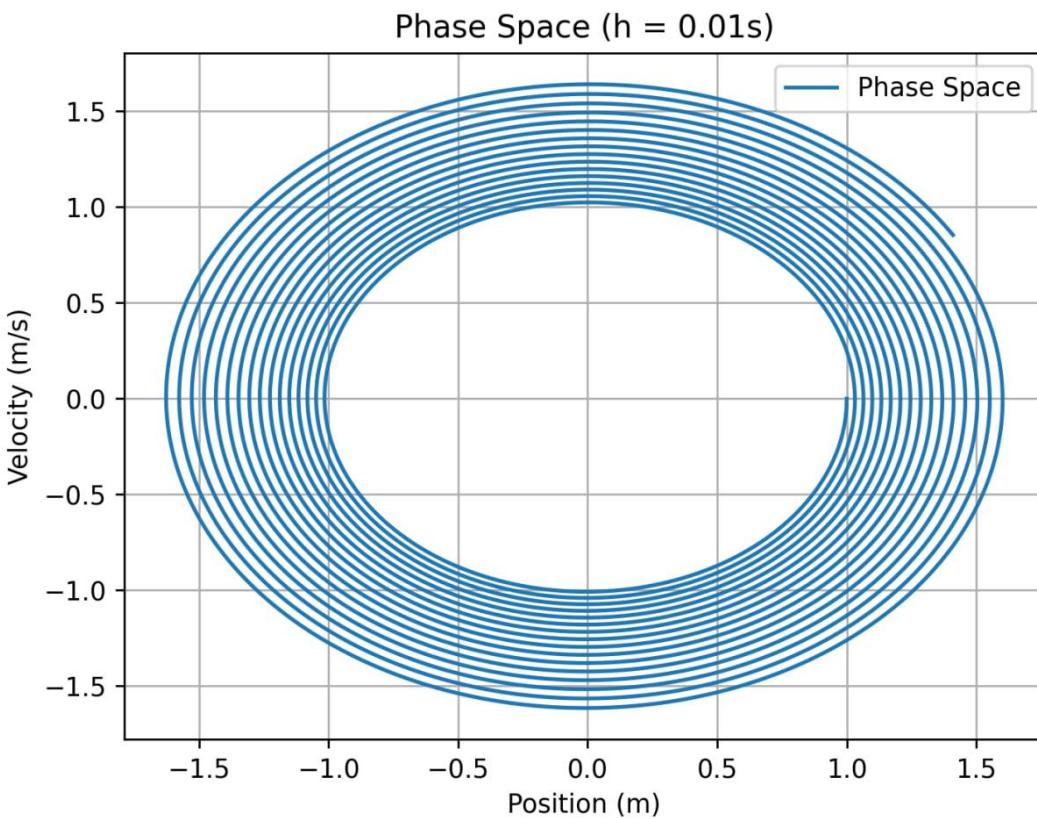
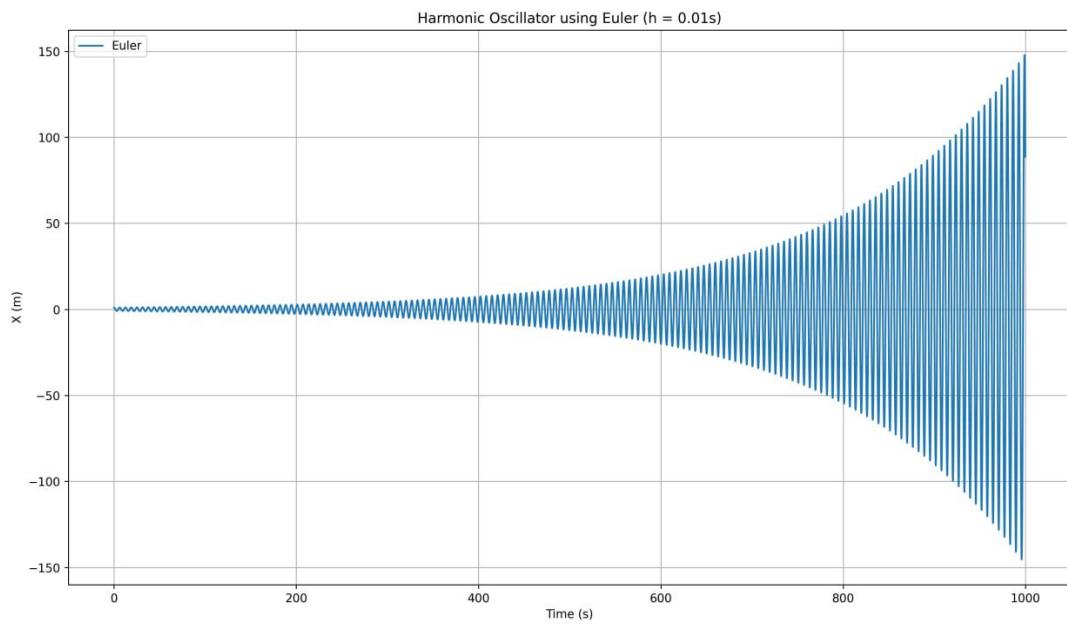


همچنین انتظار می رود سیستم پایستگی انرژی داشته باشد و نمودار آن در فضای فاز دایروی باشد.

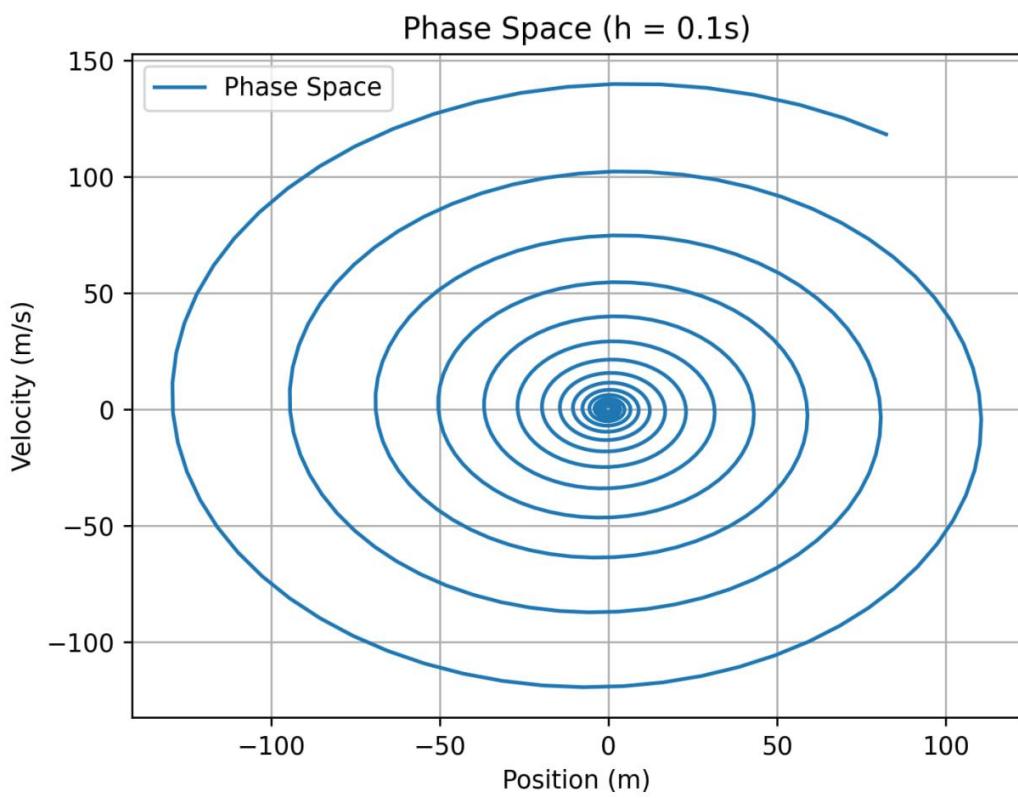
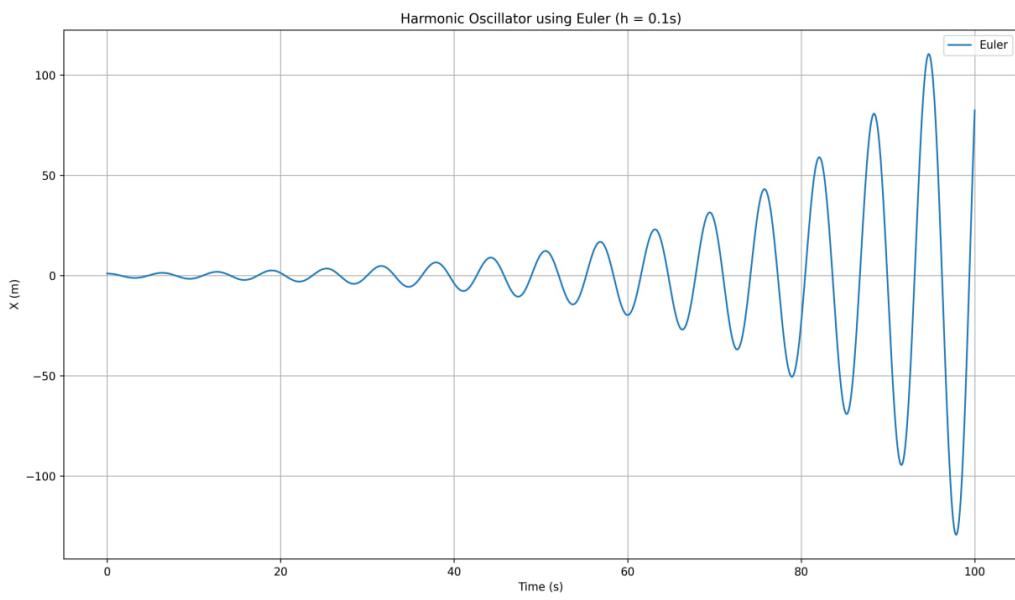
ولی با زیاد کردن زمان یا  $h$  سر و کله ناپایداری و خطای الگوریتم مشاهده می شود.



می بینیم که دامنه به مرور زمان در حال افزایش است و منحنی فضای فاز آن در حال بزرگ تر شدن است یعنی به سیستم انرژی وارد می شود.  
برای مشاهده بیشتر این موضوع باز زمان و  $h$  را افزایش می دهیم.

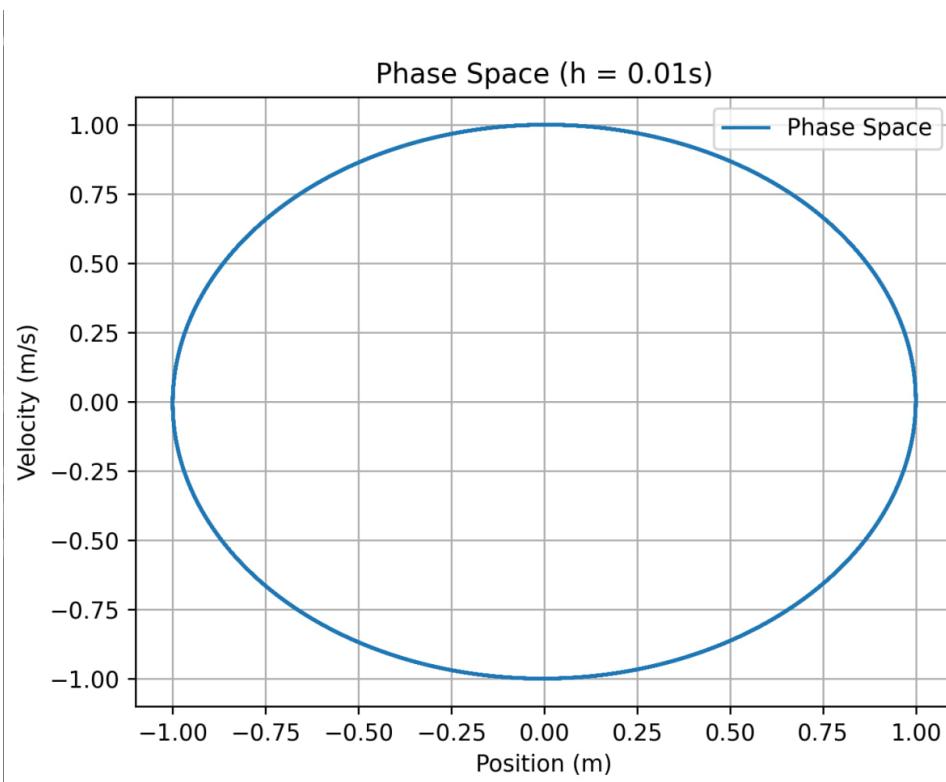
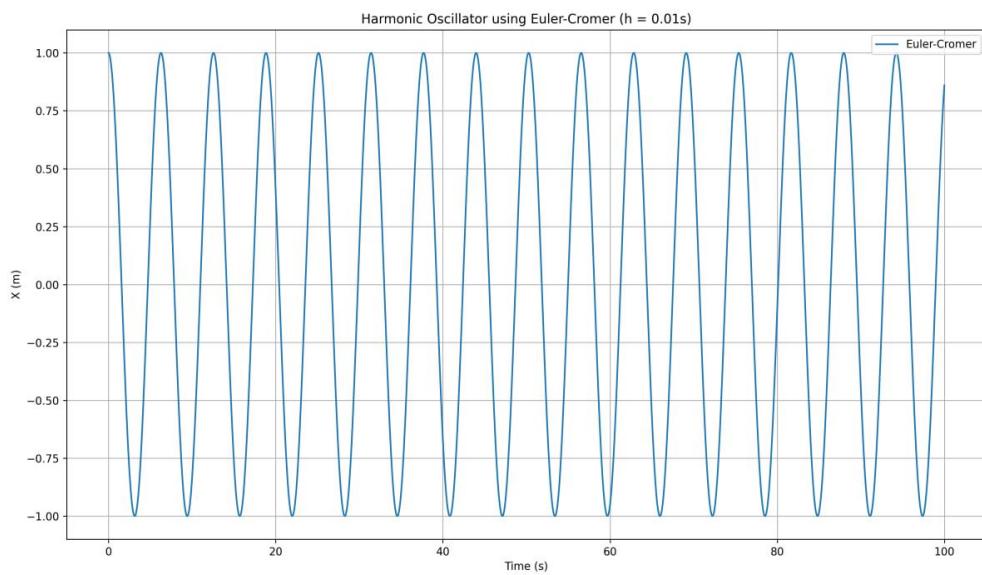


واضح دامه در حال افزایش آشوبناک (نمایی) و انرژی در حال زیاد شدن است.

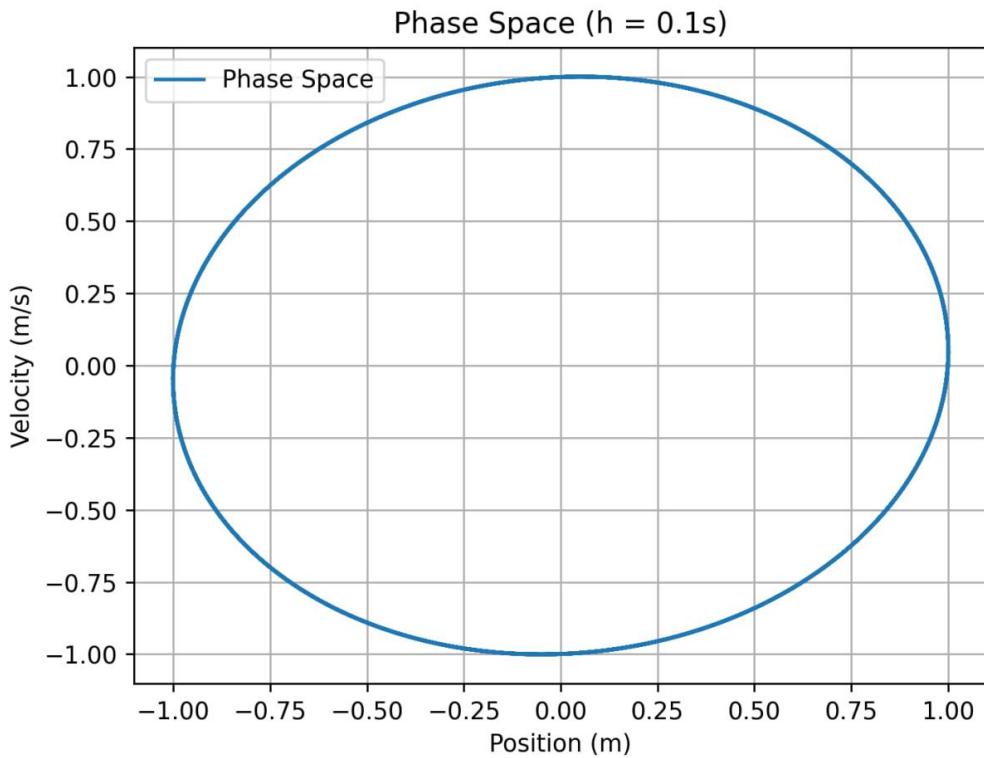
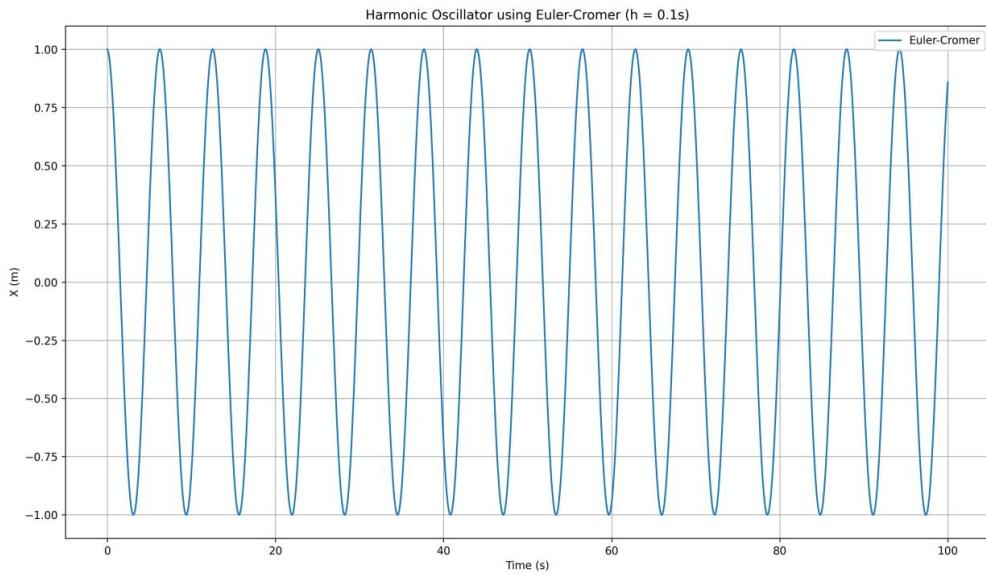


همانطور که مشاهده می شود الگوریتم اویلر بسیار دقیق نمایند و ناپایدار است.

اویلر-کرامر:

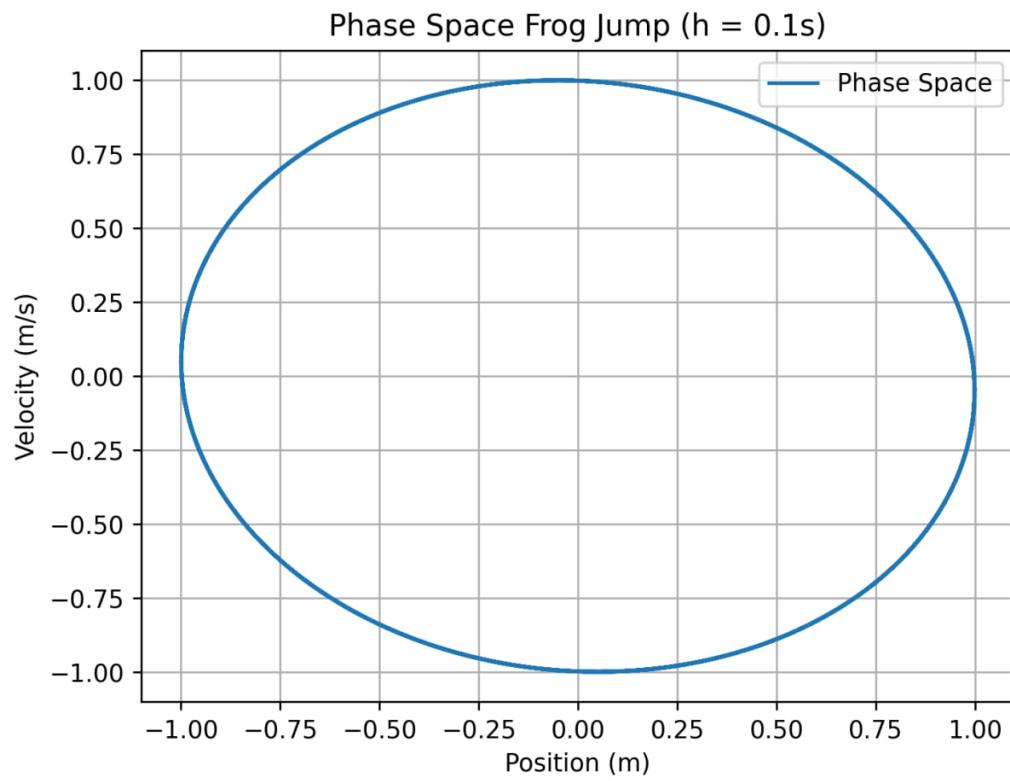
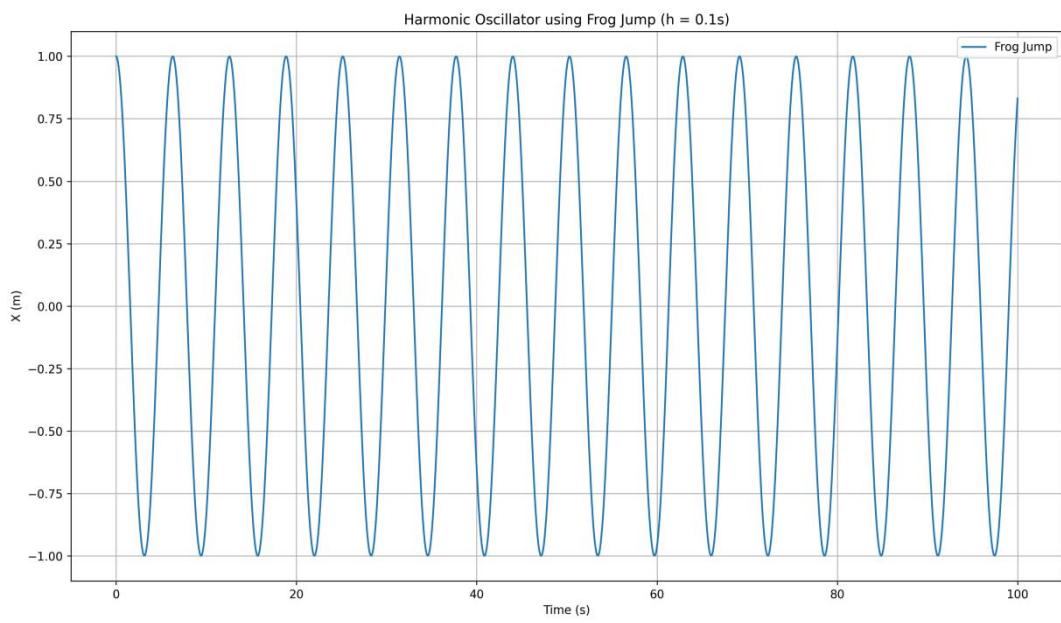


همینجا می بینیم که الگوریتم اویلر-کرامر بسیار دقیق تر از الگوریتم اویلر است و منحنی فضای فاز آن ضخامت کمتری دارد.

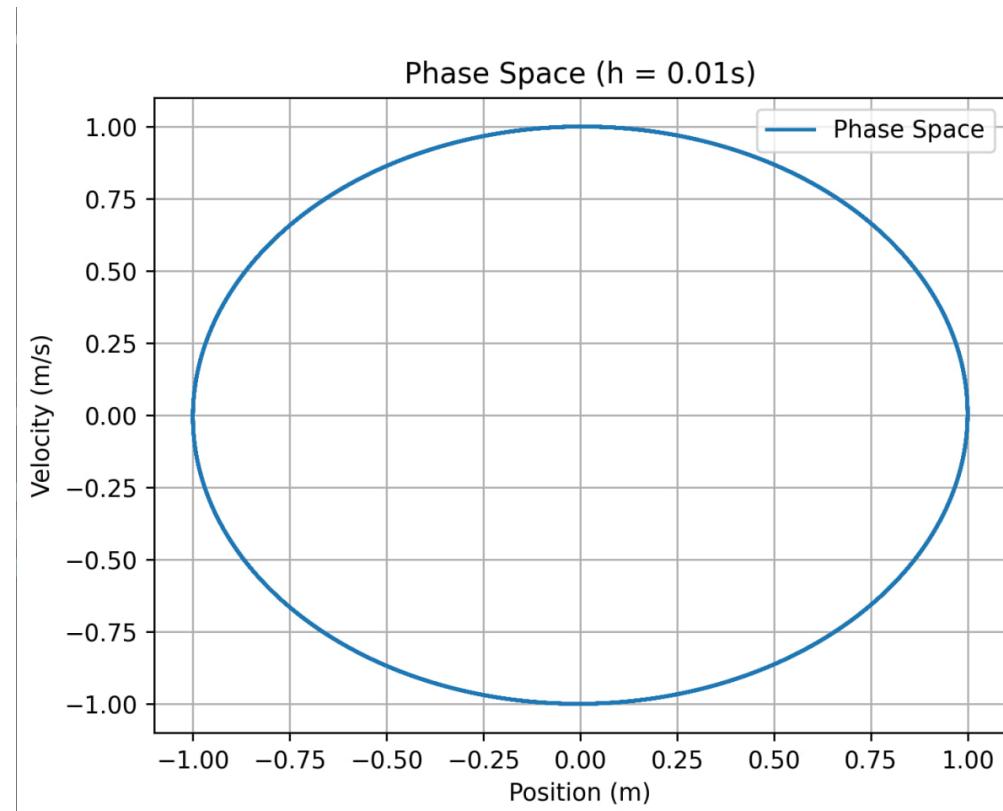
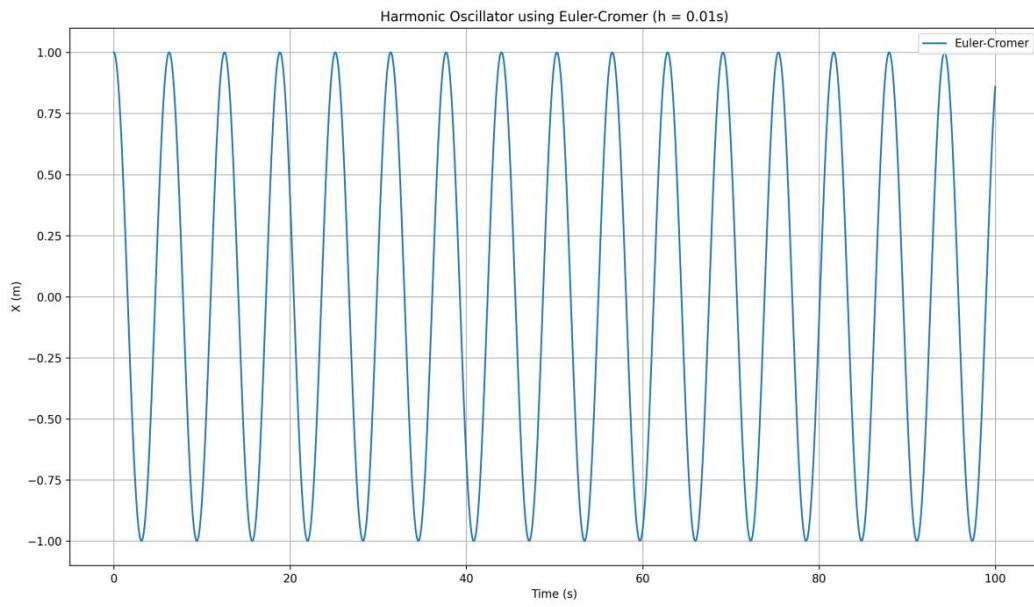


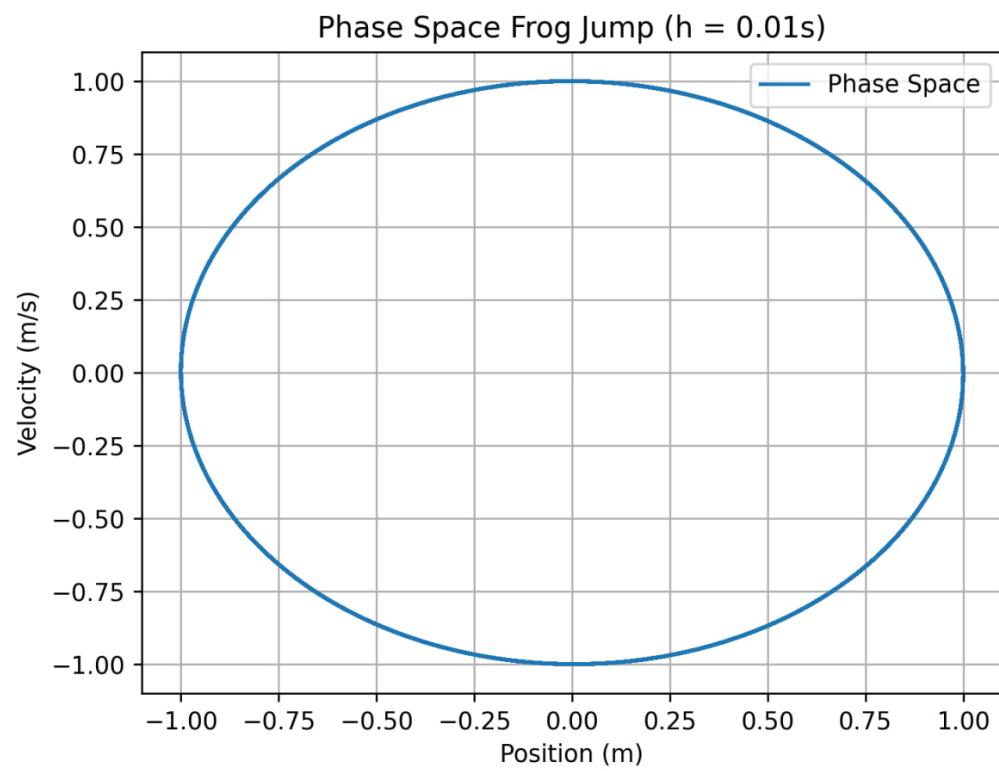
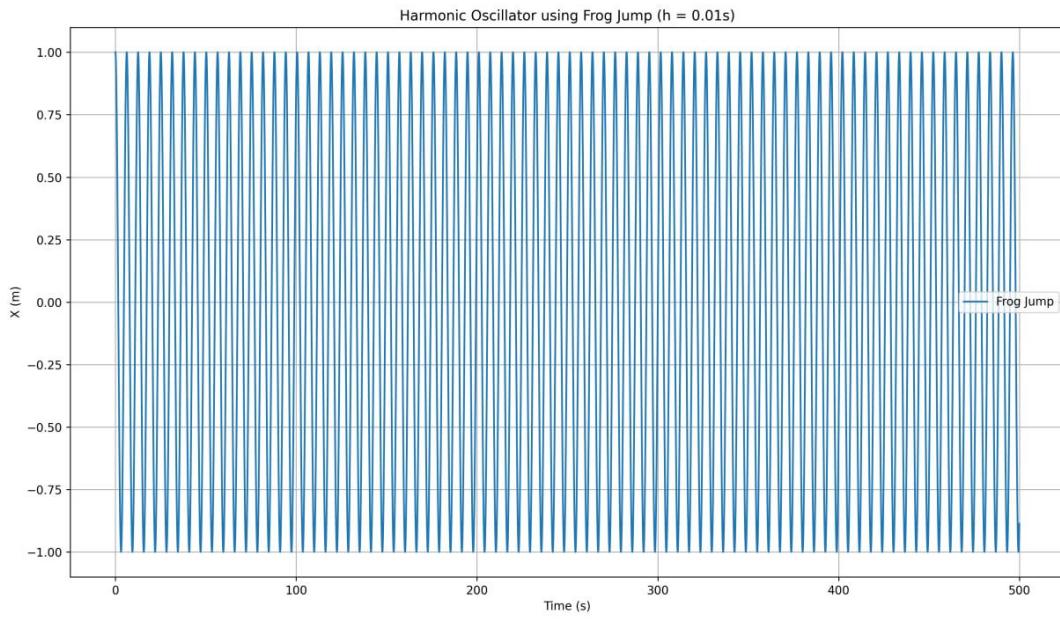
پرش قورباغه ای:

قدم اول پرش قورباغه ای را با اویلر حساب می کنیم.



پرش قورباغه ای نیز بسیار دقیق تر است و تقریبا هیچ افزایش دامنه ای مشاهده نمی شود.

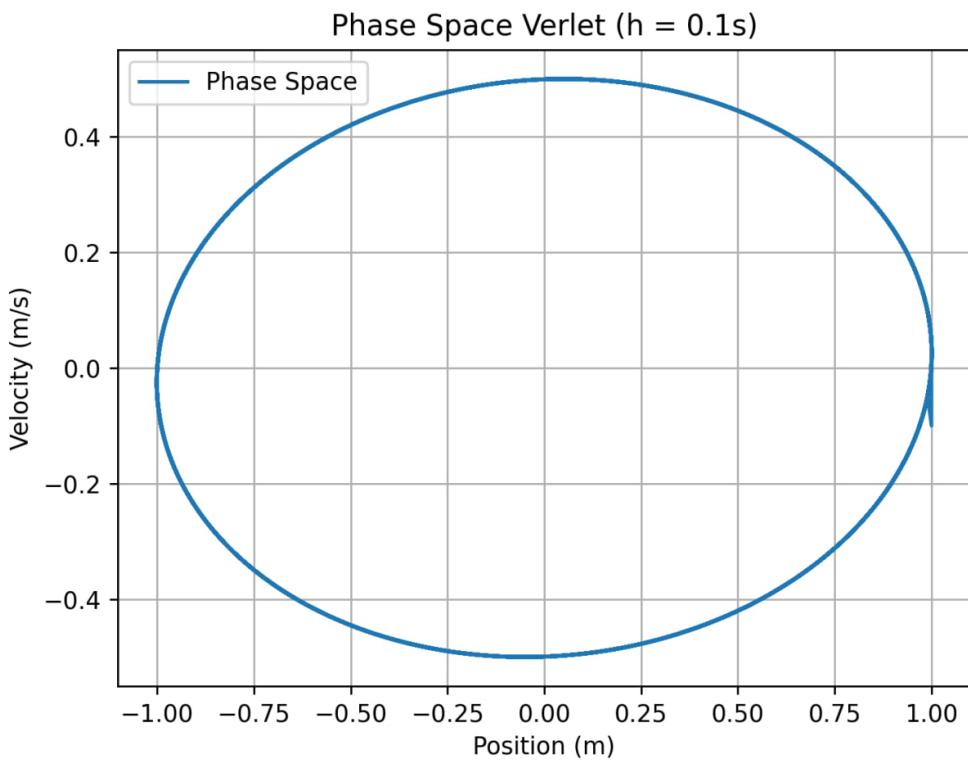
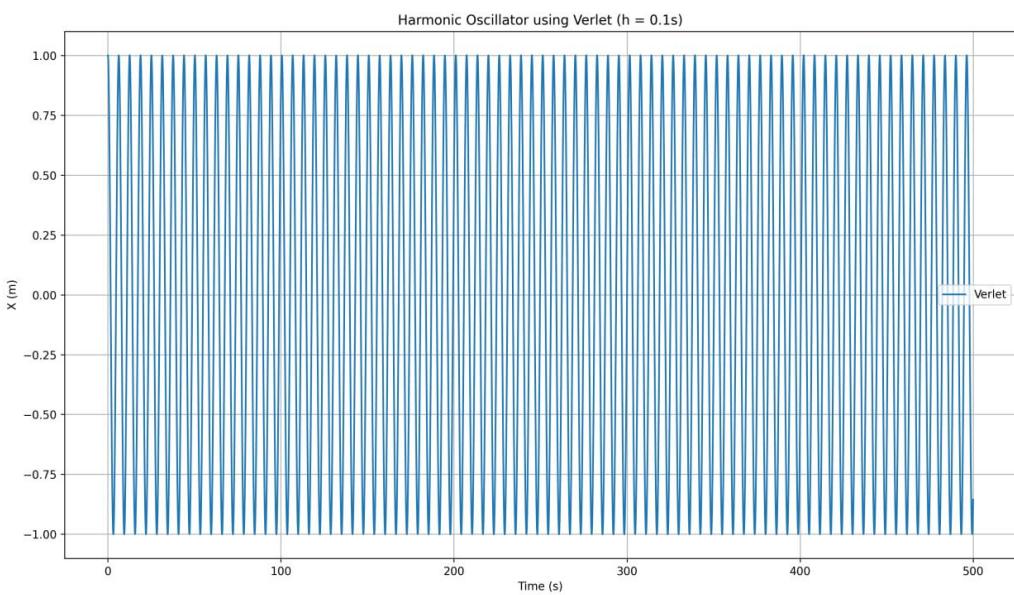




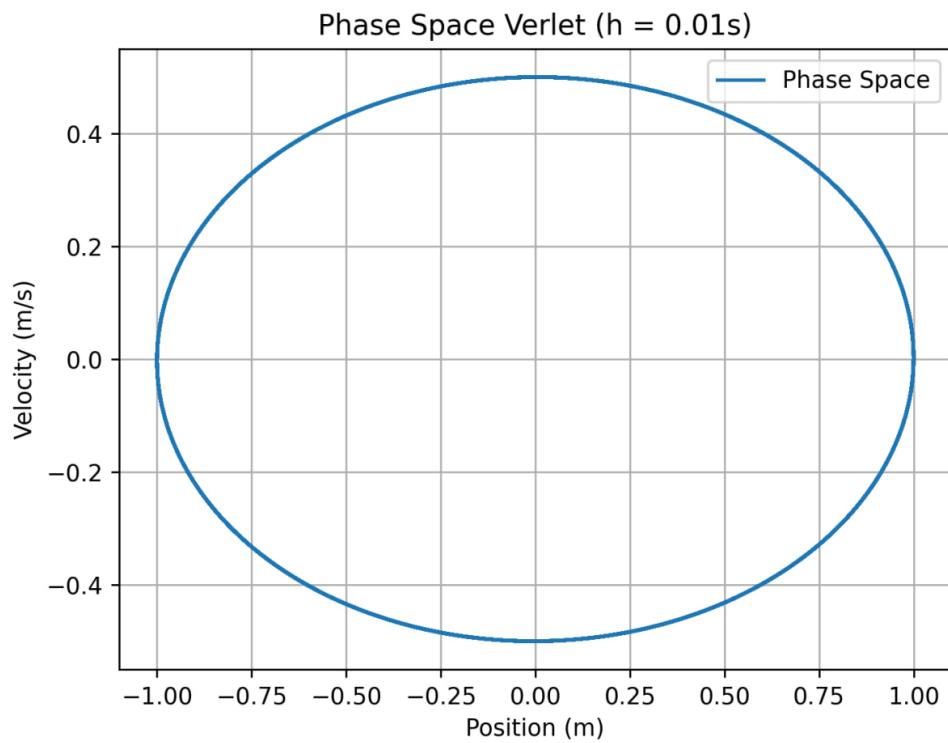
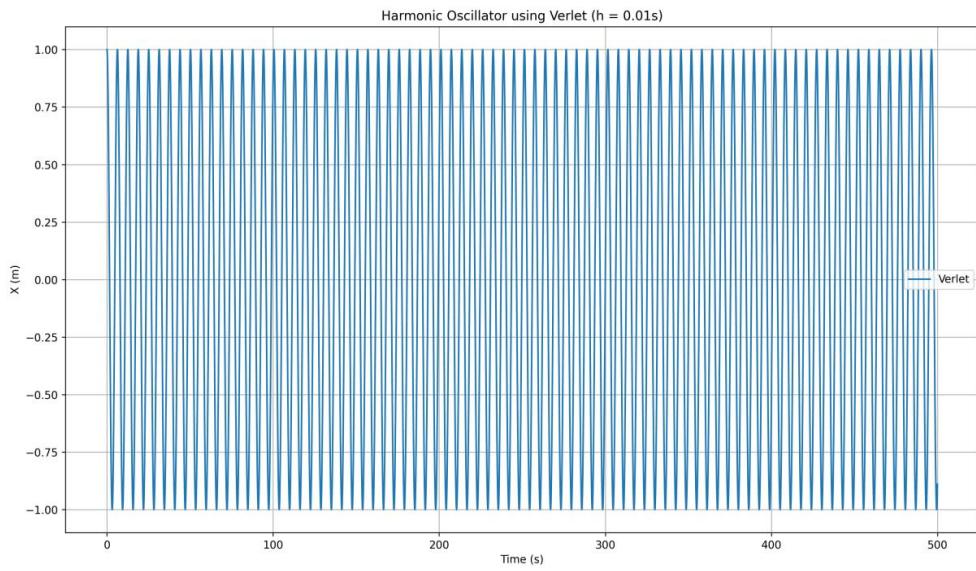
همچنان افزایش دامنه و انرژی بسیار کم است.

ورله:

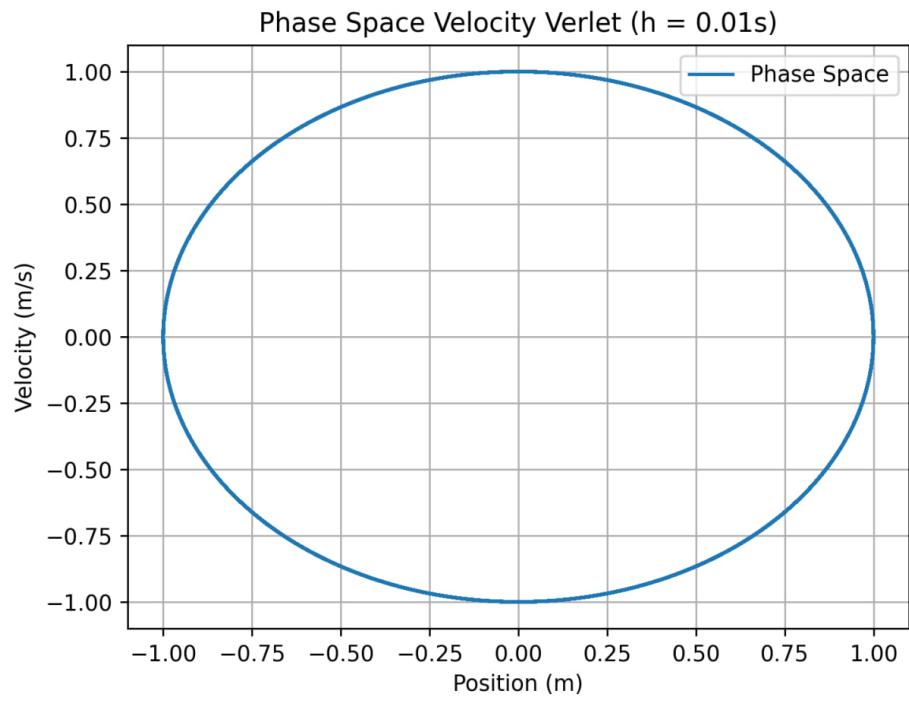
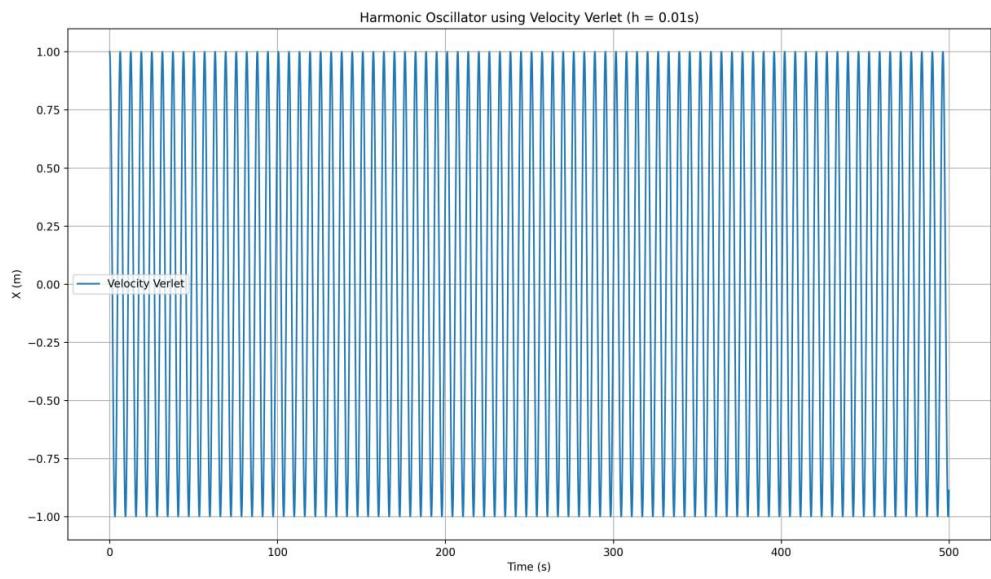
قدم دوم را با اویلر محاسبه می کنیم.

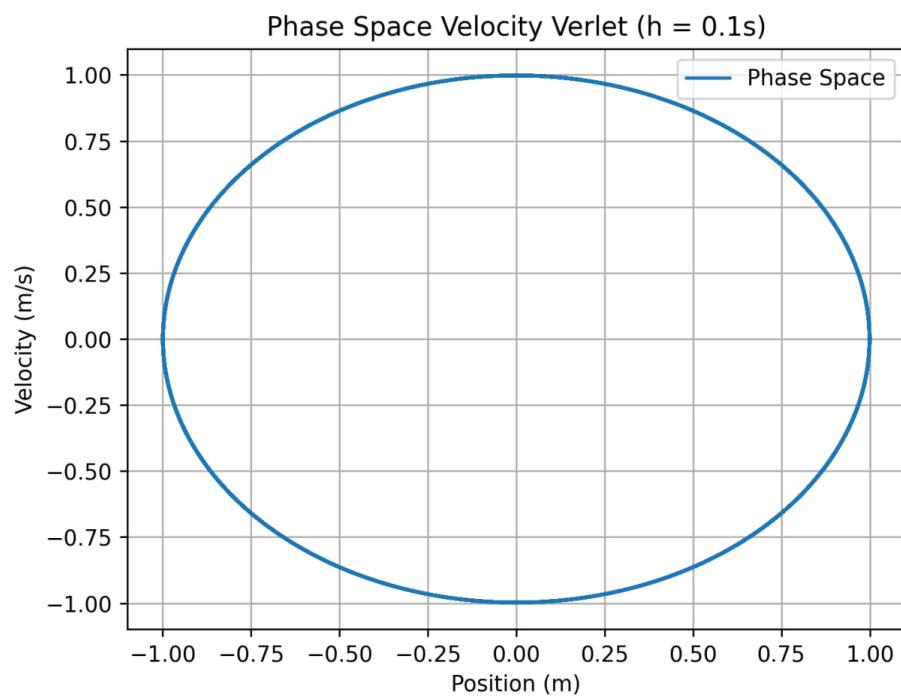
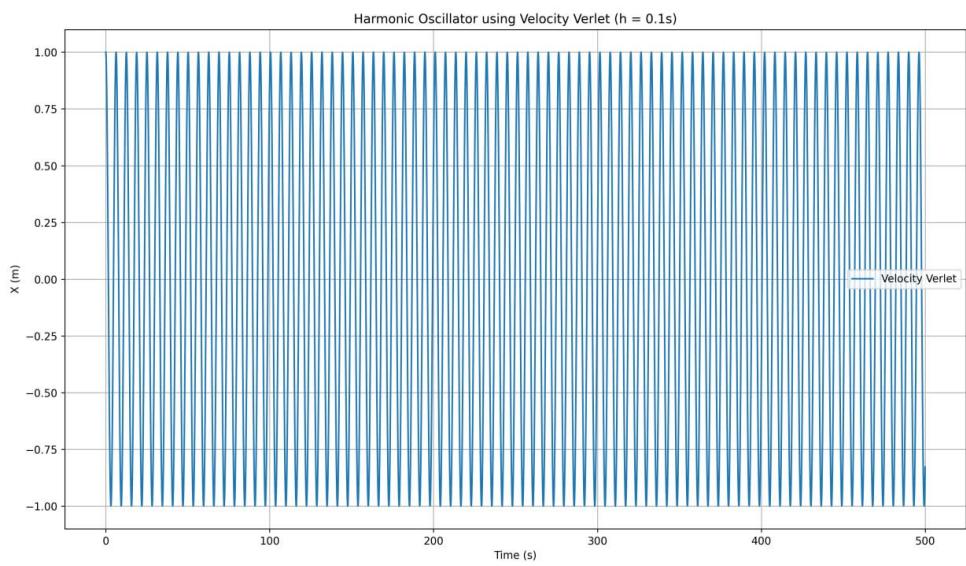


مقداری خطای نمودار فضای فاز مشاهده می شود. ولی با کاهش  $\hbar$  برطرف می شود.

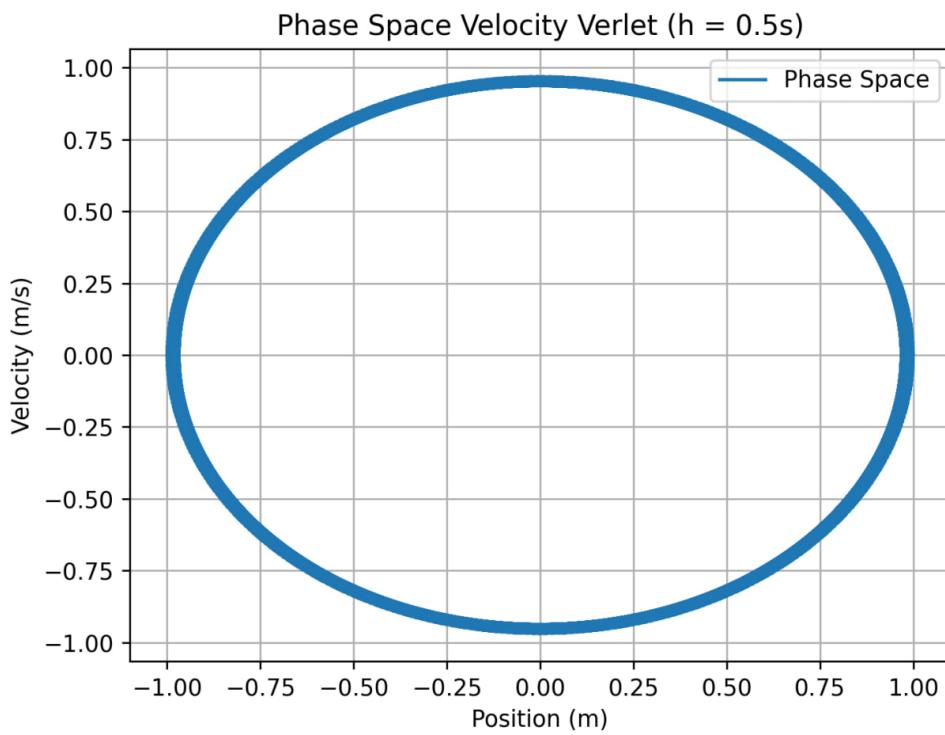
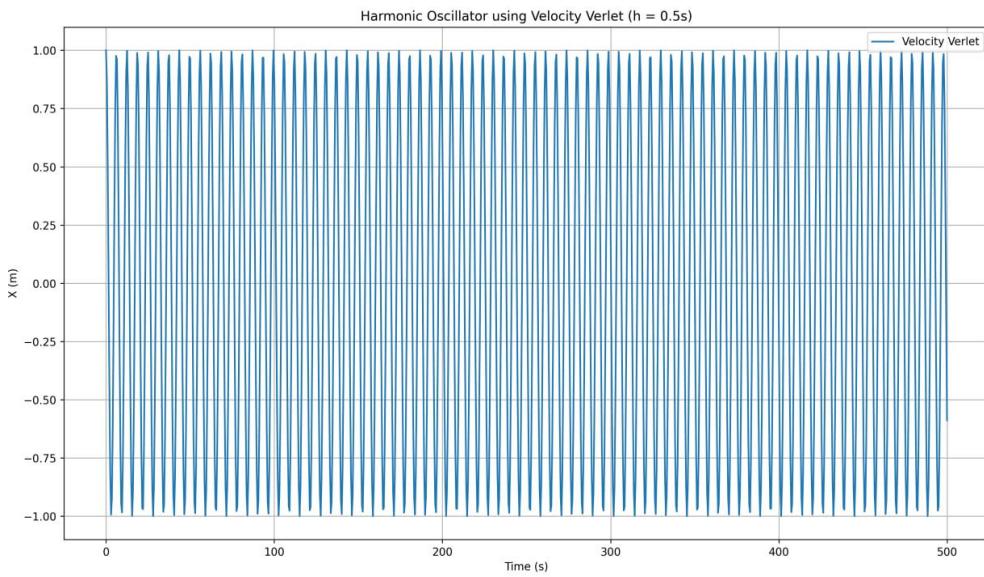


ورله سرعتی:





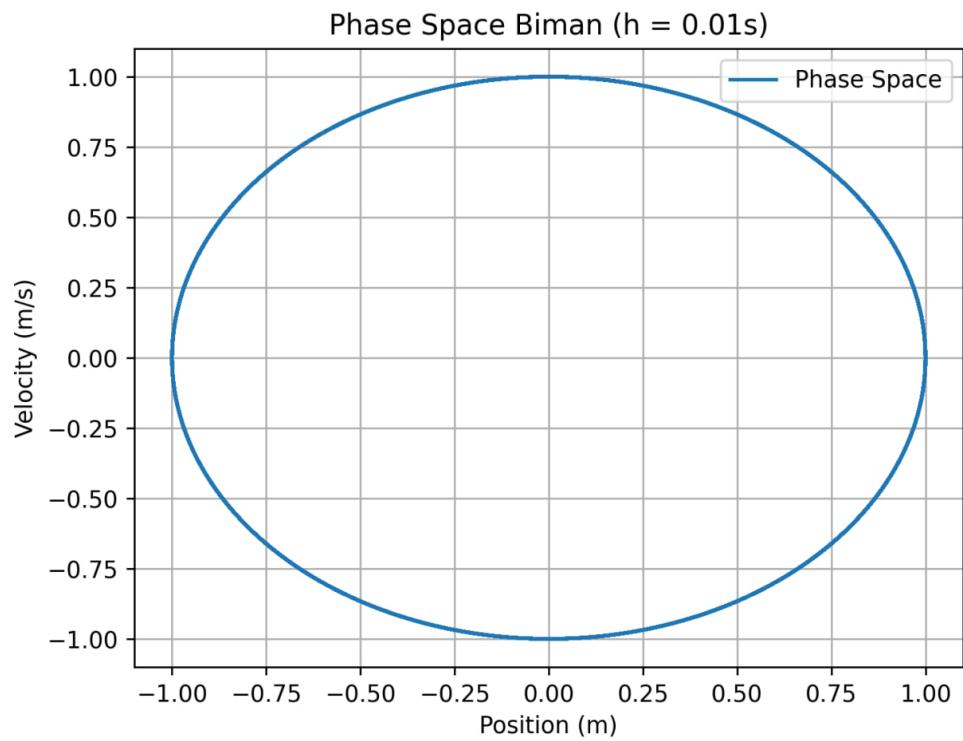
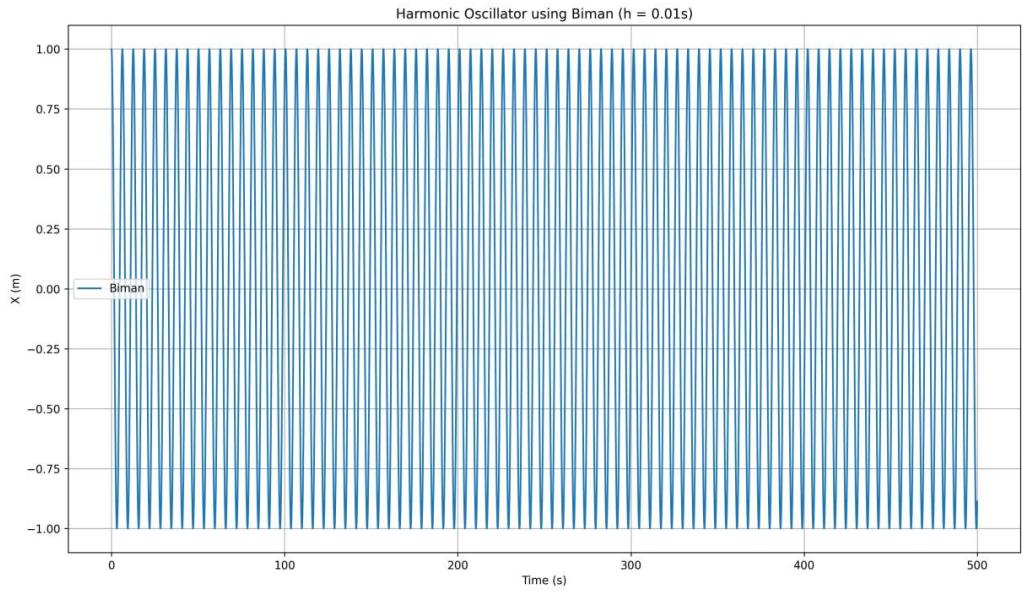
این الگوریتم بسیار دقیق و سریع است و حتی با  $h=0.1$  هم خطای کمی دارد.

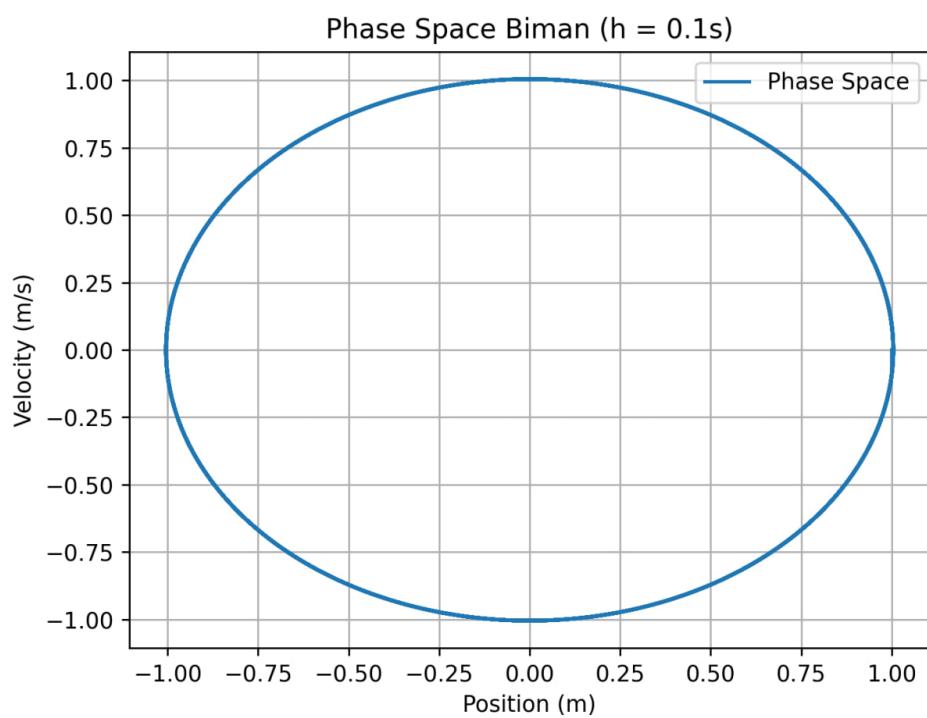
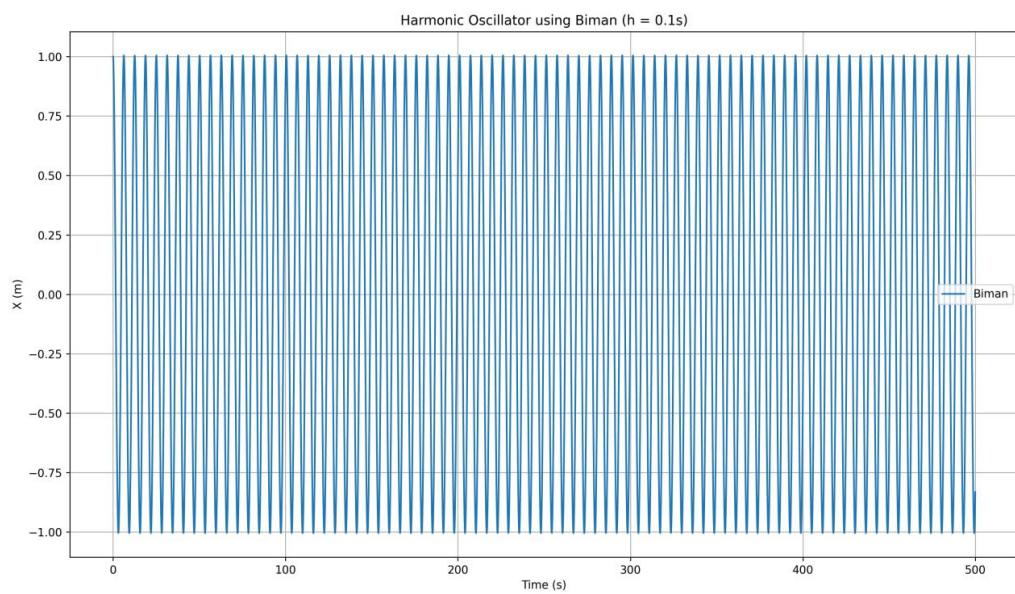


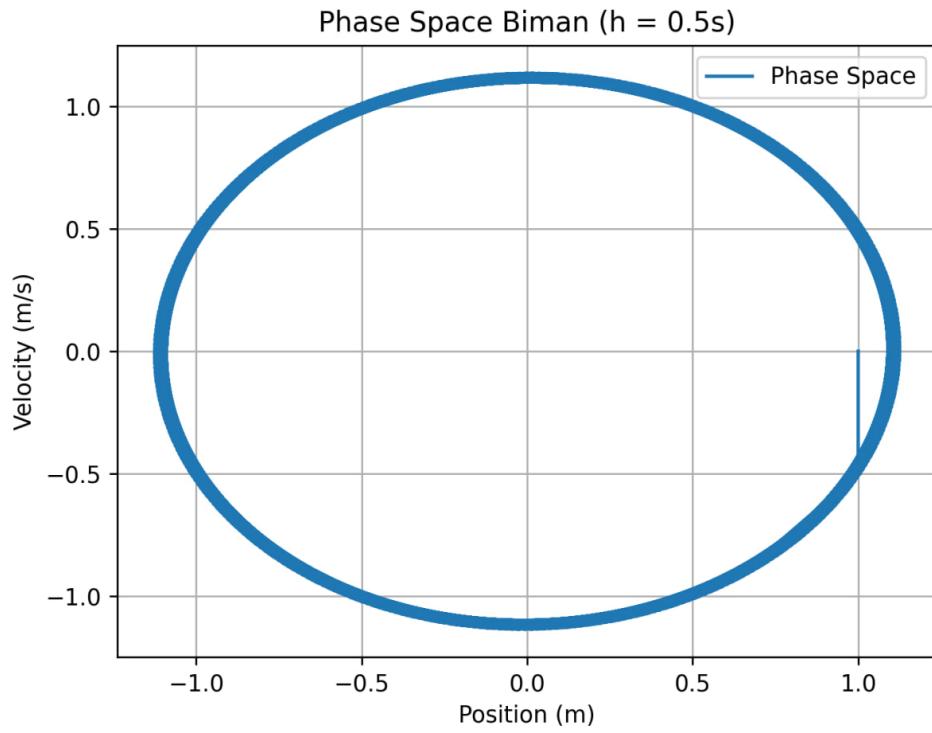
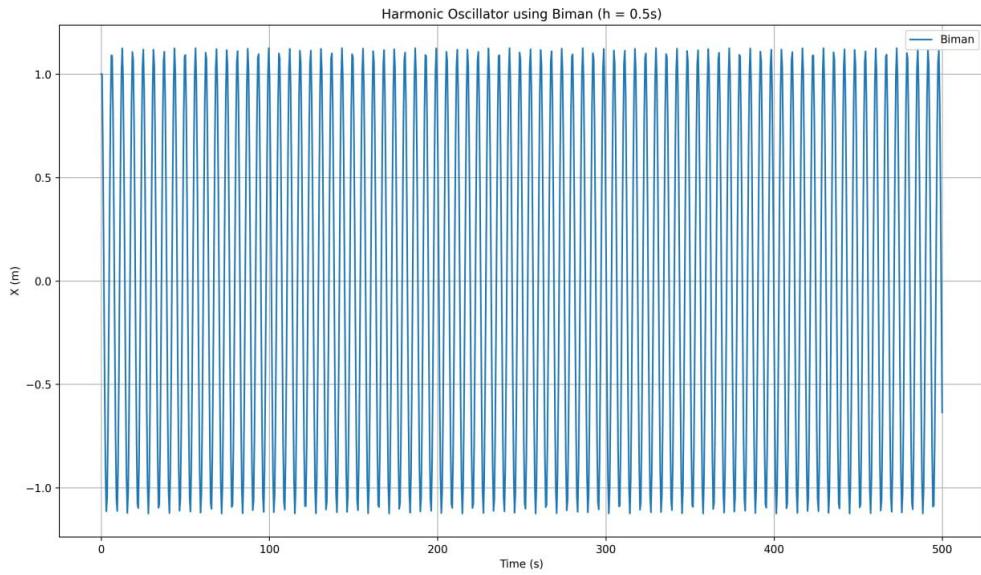
در  $h=0.5$  رفتار آشوبناک را برای دامنه و کاهش انرژی را برای فضای فاز مشاهده می کنیم.

بیمن:

قدم اول با اویلر محاسبه می شود.

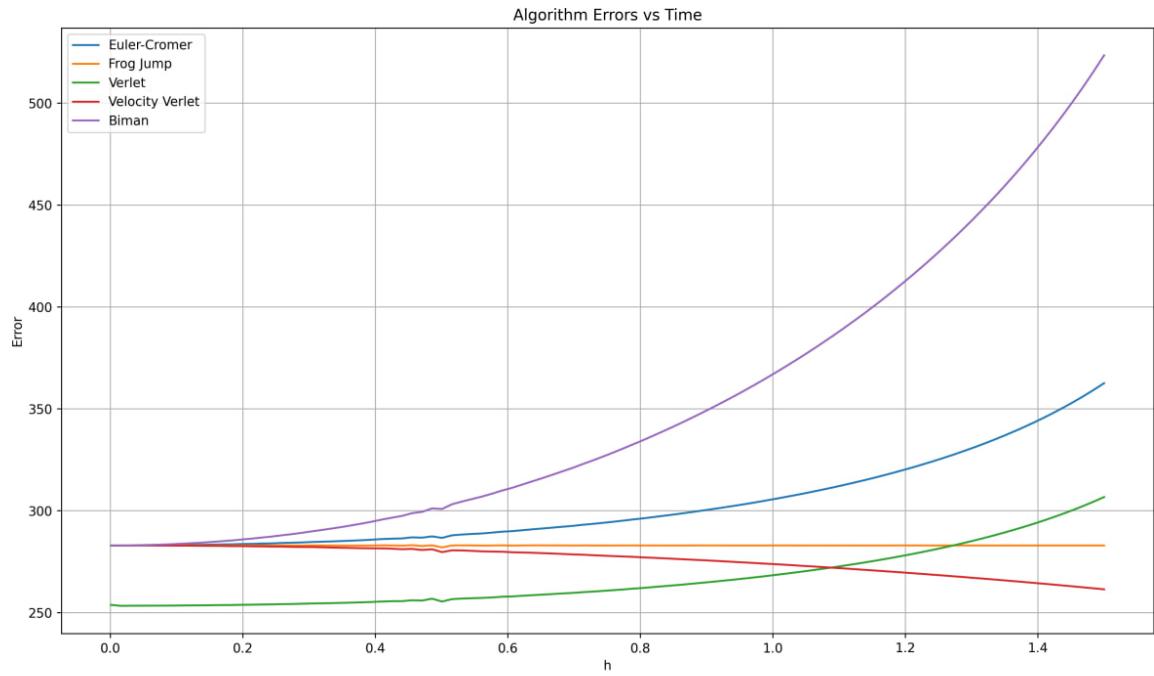




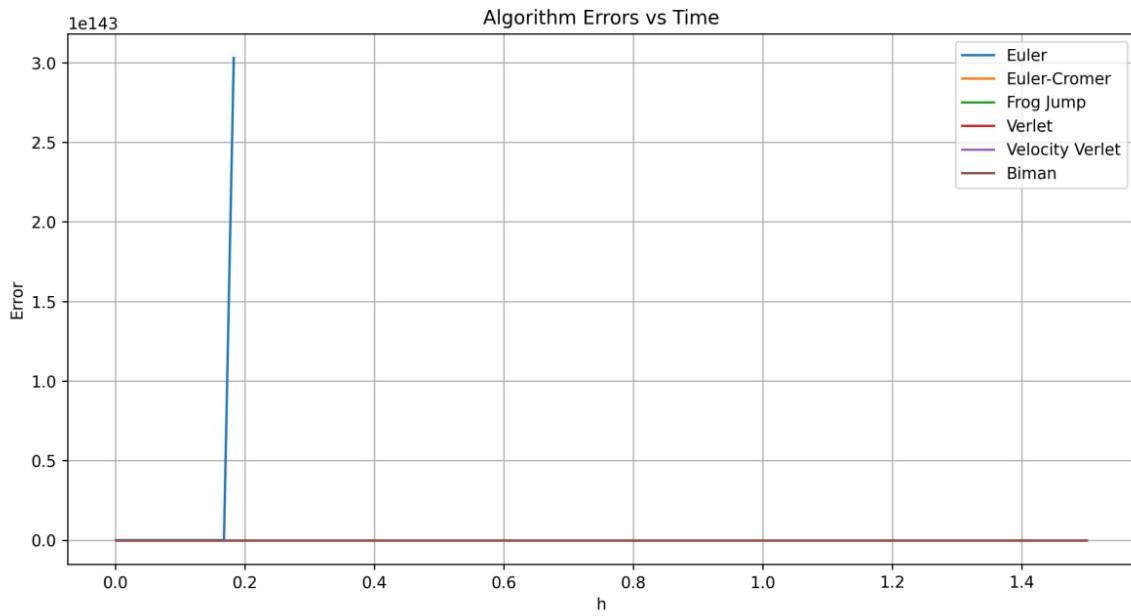


بیمن الگوریتمی نسبتاً نادقيق است و در نمودار فضای فاز خیلی با نمودار مورد انتظار تفاوت دارد و انرژی خیلی بیشتری دارد.

برای بررسی بیشتر خطای الگوریتم ها، کدی نوشتیم که فاصله منحنی فضای فاز بدست آمده را با مقدار تئوری در هر نقطه محاسبه و به عنوان خط گزارش می کند. این کار را برای  $h$  های مختلف انجام می دهیم.



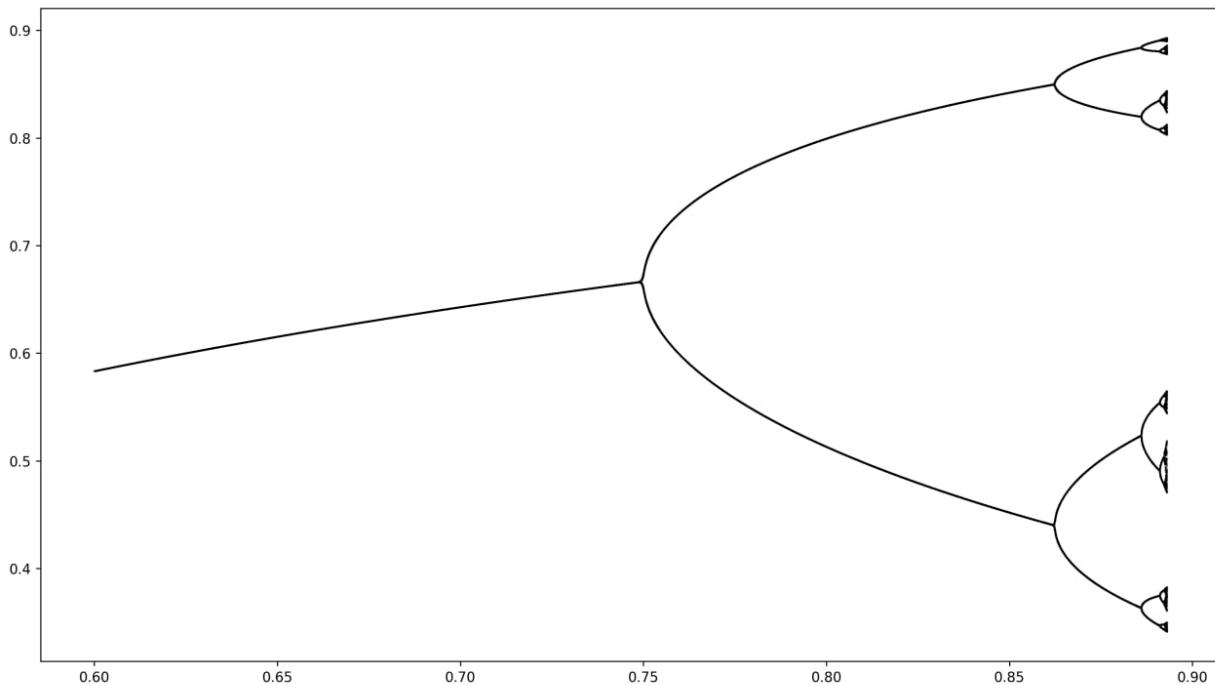
به ترتیب خطای ورله سرعتی و پرش قورباغه ای از بقیه الگوریتم ها کمتر است.



خطای الگوریتم اویلر اصلا قابل مقایسه با بقیه الگوریتم ها نبود و مرتبه بسیار بزرگ تری داشت.

### - آشوب: 3 chaos.py

در این سوال، تابع logistic\_map را تعریف می کنیم که آرایه ای از  $x$  ها که با فرمول گفته شده تولید شده اند می دهند. سپس برای 100 عدد آخر این آرایه، اعداد غیر یکسان (unique) را پیدا می کنیم. و این کار را برای 2 های مختلف تکرار می کنیم. نمودار درختی آشوب به شکل زیر می شود.



همانطور که مشاهده می شود برای 2 های مختلف هر شاخه نمودار به دو شاخه مجزا تقسیم می شود. می توان ضریب Feigenbaum constants را از نسبت فاصله این دو شاخه ها محاسبه کرد.

r	dr	Feigenbaum constants
0.7495		
0.8623	0.1128	
0.88607	0.02377	4.745477493
0.8911	0.00503	4.725646123
0.89225	0.00115	4.373913043
		mean = 4.61501222

$$Feigenbaum\ constants = 4.615$$