

دوره آموزشی برنامه متمتیکا
دوره سایت فرادرس
جلسه اول
حذف متغیر در برنامه (هر متغیری قبل از این دستور پاک می‌شود).

```
1 Clear["Global`*"]
```

تنها متغیر a حذف می‌شود.

```
1 Clear[a]
```

غیرفعال کردن یک دستور:
منوی Un/comment Section ← edit
کلید میانبر : Alt + /
نشان دادن هر متغیری (چه مقدار دهنده و چه مقدار نشده)

```
1 Clear["Name`*"]
```

- * همیشه باید در اول هر برنامه از دو کد Clear و Remove استفاده کنیم تا در محاسبات مشکلی پیش نیاید.
- * در متمتیکا حرف اول همه دستورها باید بزرگ شود.
- * روش اجرا دستورهای در یک سل : Shift + Enter
- اجرای دستورهای همه سل‌ها به صورت یکجا: Evaluation → EvaluateNotebook
- دستور برای مقداردهنده یک متغیر:

```
1 Variable_name = Value
```

برای مثال در کد زیر مقدار ۲ را درون متغیری با نام a قرار می‌دهیم:

```
1 a = 2
```

مشکلی که این نوع مقداردهنده دارد این است که بعد از اجرا سلولی که این کد درون آن است، این مقدار در خروجی چاپ می‌شود. برای اینکه هم مقدار موردنظر در متغیر موردنظر ذخیره شود و هم اینکه در خروجی نمایش داده نشود. دو روش وجود دارد. روش اول:

```
1 a = 2;
```

روش دیگری که موجود است:

```
1 a := 2
```

تفاوت این دو دستور باهم این است که در روش اول مقدار حساب و ذخیره می‌شوند. ولی در روش دوم مقدار حساب نمی‌شود تا زمانی که از آن در جایی استفاده شود

روش متوقف کردن برنامه در حال اجرا:
Local Quit kernel Evaluation
جواب سوال پرسیده شده: Quit

* هر دستور در متمتیکا باید در براکت ([]) نوشته شود:

```
1 Sin[x] , Clear[c]
```

دستورهای پایه ریاضی
برای این بخش دو متغیر از قبل تعریف می‌کنیم:

```
1 a = 5  
2 b = 8
```

دستور جمع:

```
1 a + b Out : 13
```

دستور تفریق:

```
1 a - b Out : -3
```

دستور ضرب:

```
1 a * b Out : 40
```

دستور تقسیم (روش اول):

```
1 a / b          Out : 5 / 8
```

دستور تقسیم (روش دوم):

```
1 a/b          Out : 5 / 8
```

نحوه خروجی گرفتن با دقت موردنظر:

```
1 N[expression]           e.g: N[1/3]          Out: 0.333
2 N[expression , accuracy]   e.g: N[1/3 , 10]    Out: 0.333333333333
```

برای نوشتن این روش پس از نوشتن صورت تقسیم (در این مثال a / b) را $\text{Ctrl} + \text{Shift} + =$ میزنیم.
روش تایپ کردن فرمولهای خاص:

Palettes → BasicMathAssistant

استفاده از حروف یونانی و اسمايلها در این قسمت نیز موجود است.

هز گزینه کلید میانبر مخصوص خود را دارد. برای آشنایی با کلید میانبر هر گزینه، روی گزینه کمی نگه دارید.
* خطای ۱۰۴۲: ایجاد اشکال در دستور وارد شده، مثلا:

```
1 W = W + 1      ,   W = W1 + 1
```

روش دیگر برای نوشتن این عبارت: در این حالت کنترلی در تعداد اعشار نداریم.

```
1 1 / 3 // N
```

جلسه دوم: معرف توابع
تابع رادیکال:

```
1 Sqrt[4]           Out: 2
2 Sqrt[x**2]       Out: √(x^2)
```

: (Simplify) تابع ساده سازی

```
1 Simplify[% , x > 0]          Out: +x
2 Simplify[% , x < 0]          Out: -x
```

در این مثال نماینده آخرین خروجی است.
تابع نمایی:

```
1 Exp[x]           Out: e**x      e = 2.7
2 Exp[2]           Out: e**2
3 Exp[2] // N      Out: 7.38906
```

تابع جزء صحیح:

```
1 Floor[2.3]        Out: 2
```

تابع قدرمطلق:

```
1 Abs[-2]           Out: 2
2 Abs[-x]           Out: Abs[x]
```

* در مثال دوم این کد، خروجی از تابع بیرون در نمی آید چون x میتواند منفی و چیزهای دیگر باشد.
تابع علامت:

```
1 Sign[{-2, 0, 3}]        Out: {-1, 0, 1}
```

تابع فاکتوریل (روش اول):

```
1 x = 5
2 Facorial[x]         Out: 120
```

تابع فاکتوریل (روش دوم):

```
1 x!                  Out: 120
```

تابع لگاریتم:

```
1 Log[x]             Out: Log[x]
```

زمانی که برای الگوریتم پایه تغیریف نکنیم، نرم افزار به صورت خودکار پایه را عدد نپر ($\ln x$ یا E) قرار می‌دهد.

```
1 Log[E]
```

```
Out: 1
```

تغیریف پایه الگوریتم:

```
1 Log[a, x] a is The base of the logarithm  
2 Log[10, 1000] Out: 3
```

توابع مثلثاتی:

```
1 Sin[x] e,g: Sin[Pi/3] Out: sqrt[3] / 2  
2 Cos[x]  
3 Tan[x]  
4 Cot[x]  
5 ArcSin[x]  
6 ArcTan[x]
```

به صورت دیفالت مقدار محاسبه این توابع به صورت رادیان است، اگر بخواهیم نرم افزار حاصل درجه موردنظر را حساب کند، این گونه عمل می‌کنیم:

```
1 Sin[30 Degree] Out: sqrt[3] / 2  
2 Sin[30 Degree] // N Out: 0.5
```

توابع هیپربولیک

```
1 Sinh[x]  
2 Cosh[x]  
3 Coth[x]
```

تابع تولید اعداد تصادفی
برای اعداد حقیقی:

```
1 RandomReal[{-2, 2}, 5] Out: {1.31496, 0.0146972, -1.486, -1.93584, 0.638742}  
2 RandomReal[{-2, 2}, {2, 5}] Out: {{1.86401, -1.47587, 0.0531725, 1.73636, 0.243321},  
3 {1.50354, 0.924516, 1.60522, 0.870081, -1.22427}}
```

در مثال اول پنج عدد تصادفی در بازه $[-2, 2]$ -انتخاب می‌شوند.
در مثال دوم دو دسته پنج تایی عدد تصادفی در بازه $[-2, 2]$ -انتخاب می‌شوند.
برای اعداد صحیح:

```
1 RandomInteger[{-2, 2}, 5] Out: {-2, 0, 1, 2, -2}
```

پنج عدد صحیح تصادفی در بازه $[-2, 2]$ -انتخاب می‌شوند.
تبديل عدد به عامل های اول:

```
1 FactorInteger[10] Out: {{2, 1}, {5, 1}} 10 = 2^1 * 5^1
```

به توان رساندن:

```
1 Superscript[2, 3] Out: Superscript[2,3]  
2 Superscript @@ {2, 3} Out: Superscript[2,3]  
3 Superscript @@ {{2, 3}, {6, 5}} Out: {Superscript[2,3], Superscript[6,5]}  
4 Superscript @@ {{2, 3}, {4, 5}} Out: Superscript[{2, 3}, {4, 5}]
```

```
1 CenterDot[x, y] Out: x\[CenterDot]y  
2 CenterDot @@ (Superscript @@ (FactorInteger[15])) Out: Superscript[3,1]\[CenterDot]  
Superscript[5,1]
```

کوچکترین مضرب مشترک

```
1 LCM[5, 6]
```

بزرگترین تقسیم الیه مشترک

```
1 GCD[15, 9]
```

باقي مانده

```
1 Mod[15 , 2]
```

خارج قسمت

```
1 Quotient[15, 3]
```

ترکیب

```
1 Binomial[15, 3]
```

دلtar دیراک

```
1 DiracDelta[x]
2 DiracDelta[x] // TraditionalForm Out: \[Delta](x)
```

دلتا کراینیکر

```
1 DiracDelta[m, n] // TraditionalForm
```

اعداد مختلف:

```
1 z = x + I y Out: x + I y
2 Re[z] Out: -Im[y] + Re[x]
```

برای تعریف کردن قسمت موهومی و واقعی اعداد مختلف:

```
1 Refine[Re[z], Element[{x, y}, Reals]] Out: x // Real Part
2 Refine[Im[z], Element[{x, y}, Reals]] Out: y // Imaginary Part
```

مزدوج گیری (مختلط):

```
1 Conjugate[z] Out: Conjugate[x] - I Conjugate[y]
```

ریفاین کردن

```
1 Refine[Conjugate[z], Element[{x, y}, Reals]] Out: x - I y
```

تعریف تابع دلخواه:

```
1 F[x_] = x^2 + 1 Out: 1 + x^2
2 F[3] Out: 10
3 Map[F, {2, 3}] Out: {5, 10}
4 F /@ {2, 3} Out: {5, 10}
```

برای تعریف توابع چند متغیره:

```
1 G[x_, y_] = x + y + 2 Out: 2 + x + y
2 G[{2, 3}, {5, 6}] Out: {9, 11}
```

جلسه سوم : محاسبات جبری و مثلثاتی، سری‌ها بسط یک عبارت

```
1 (x + 1)^3 Out: (1 + x)^3
2 Expand[(x + 1)^3] Out: 1 + 3 x + 3 x^2 + x^3
3 (x + 1)^3 // Expand Out: 1 + 3 x + 3 x^2 + x^3
4 Expand[(a + b)^2*(x + 1)^2] Out: a^2 + 2 a b + b^2 + 2 a^2 x + 4 a b x + 2 b^2 x
5 + a^2 x^2 + 2 a b x^2 + b^2 x^2
```

همانطور که مشاهده می‌شود در مثال آخر کدهای بالا هر دو جمله بسط داده می‌شوند. گاهی نیاز است که تنها یکی از این جملات بسط داده شود. برای این کار:

```
1 Expand[(a + b)^2*(x + 1)^2, x] Out: (a + b)^2 + 2 (a + b)^2 x + (a + b)^2 x^2
```

همانطور که مشاهده می‌شوند فقط جمله‌ای که در آن x وجود دارد، بسط داده خواهد شد.

فاکتورگیری

```
1 Factor[x^4 + x^2 + x] Out: x (1 + x + x^3)
2 Factor[x^4 + x^2] Out: x^2 (1 + x^2)
```

به مثال زیر توجه کنید:

```

1 y = x^4 + x^2 + x
2 x = a + b
3 Factor[y]      Out: (a + b) (1 + a + a^3 + b + 3 a^2 b + 3 a b^2 + b^3)

```

همانطور که مشاهده می شود در این کد، از $(a+b)$ فاکتور گرفته می شود.
ساده کردن یک عبارت کسری

```

1 Cancel[(x^2 - 1)/(x - 1)]      Out: 1 + x

```

تجزیه کسر به کسرهای جزئی

```

1 Apart[1/((1 + x)*(5 + x))]   Out: 1/(4 (1 + x)) - 1/(4 (5 + x))
2 Expand[(1 + x)*(5 + x)]       Out: 5 + 6 x + x^2
3 Apart[1/%]                     Out: 1/(4 (1 + x)) - 1/(4 (5 + x))

```

بسط عبارات مثلثاتی و تبدیل آنها به عبارت نهایی

```

1 Cancel[Sin[2*x]/Sin[x]]        Out: Csc[x] Sin[2 x]
2 Cancel[Sin[2*x]/Sin[x], Trig -> True]  Out: 2 Cos[x]

```

ساده سازی عبارت های مثلثاتی

```

1 TrigFactor[Cos[x + y] + Sin[x]*Sin[y]]      Out: Cos[x] Cos[y]

```

بسط عبارت های مثلثاتی

```

1 TrigExpand[Cos[x + y]]           Out: Cos[x] Cos[y] - Sin[x] Sin[y]
2 TrigExpand[Sin[2*x]]             Out: 2 Cos[x] Sin[x]

```

تبدیل عبارت های مثلثاتی درجه های بالاتر به عبارت خطی

```

1 TrigReduce[2*(Cos[x])^2]         Out: 1 + Cos[2 x]
2 TrigReduce[(Cos[x])^3]           Out: 1/4 (3 Cos[x] + Cos[3 x])

```

تبدیل عبارت های مثلثاتی به نمایی و بر عکس

```

1 TrigToExp[Cos[x]]               Out: E^(-I x)/2 + E^(I x)/2
2 ExpToTrig[Exp[Ix]]              Out: Cosh[Ix] + Sinh[Ix]

```

ساده سازی عبارت ها

نکته: در نرم افزار متمتیکا در دو حالت زیر ضرب تعریف می شود:

(۱) بین دو عبارت علامت ضرب "*" قرار گیرد.

(۲) بین دو عبارت اسپیس گذاشته شود.

```

1 Simplify[(x - 1) (x + 1) (x^2 + 1) + 1]      Out: x^4
2 Simplify[(Sin[x])^2 + (Cos[x])^2]                Out: 1
3 FullSimplify[Cosh[x] - Sinh[x]]                  Out: E^-x

```

نکته ۱: Simplify از FullSimplify قوی تر است. کارایی هر دو یکی است.

نکته ۲: گاهی باید برای ساده سازی یک عبارت، ابتدا باید آن Expand شود یا در کسرها از Cancel استفاده شود.
سری (مجموعه ها)

```

1 Sum[Sin[i x], {i, 1, 5}]      Out: Sin[x] + Sin[2 x] + Sin[3 x] + Sin[4 x] + Sin[5 x]
2 Sum[i, {i, 1, n}]              Out: 1/2 n (1 + n)
3 Sum[i^2, {i, 1, n}]            Out: 1/6 n (1 + n) (1 + 2 n)
4 Sum[Sin[i x], {i, 1, 10, 2}]   Out: Sin[x] + Sin[3 x] + Sin[5 x] + Sin[7 x] + Sin[9 x]

```

روش دیگر: با استفاده از کلید میان ESC + Sum + ESC می توان این نماد را رسم کرد.

$$\sum_{i=1}^n i$$

$$\sum_{i=1}^5 i^2$$

خروجی کدهای تصویر بالا:

```
1 Out1: 1/2 n (1 + n)
2 Out2: 55
```

حاصل ضرب:

```
1 Product[Sin[i x], {i, 1, 10}]      Out: Sin[x] Sin[2 x] Sin[3 x] Sin[4 x] Sin[5 x]
2                                Sin[6 x] Sin[7 x] Sin[8 x] Sin[9 x] Sin[10 x]
3
4 Product[Sin[i x], {i, 1, 10, 2}]  Out: Sin[x] Sin[3 x] Sin[5 x] Sin[7 x] Sin[9 x]
```

روش دیگر: با استفاده از کلید میان ESC + Prod + ESC می‌توان این نماد را رسم کرد.

خروجی کد تصویر بالا:

```
1 Out: 120
```

جلسه چهارم : حد، مشتق و انتگرال
بخش اول - محاسبه حد

```
1 Limit[f[x], x -> x0]
2 Limit[f[x], x -> x0] // TraditionalForm
```

خروجی کد بالا به شکل زیر می‌باشد:

مثالی دیگر:

```
1 Limit[Sin[x]/x, x -> 0]      Out: 1
```

نکته: اگر یک عبارت حد چپ و راست برابر نداشته باشد، حد ندارد و در نرم‌افزار پیامی طبق، این مفهوم می‌آید.

```
1 Limit[1/x, x -> 0]          Out: Indeterminate
```

همچنین می‌توانیم حد چپ و راست را به صورت جداگانه برای یک عبارت حساب کنیم. به مثال زیر توجه کنید.
حد چپ: از سمت مقادیر کمتر از صفر به سمت صفر

```
1 Limit[1/x, x -> 0, Direction -> 1]      Out: -Infinity
```

حد چپ: از سمت مقادیر بیشتر از صفر به سمت صفر

```
1 Limit[1/x, x -> 0, Direction -> -1]      Out: Infinity
```

بخش دوم - مشتق
برای انجام عمل مشتق در برنامه می‌توان به دو صورت زیر اقدام کرد:

```
1 Derivative[n][f][x + 1]
2 D[f[x], {x, n}]
```

که خروجی کد بالا به شکل زیر در ترمینال نمایش داده خواهد شد:

```
Out[•]= f^(n) [1 + x]
```

```
Out[•]= f^(n) [x]
```

نکته: در کد دوم پارامتر $f(x)$ تابع مورد نظر برآش مشتق گرفتن، پارامتر x متغیری که می خواهیم طبق آن مشتق گیری انجام دهیم و پارامتر n تعداد مشتق است.
به مثال زیر توجه کنید:

```
1 D[(Sin[x])^10, x] Out: 10 Cos[x] Sin[x]^9
```

در این مثال تابع $\sin(x)$ به عنوان تابع ما و x به عنوان پارامتر مشتق گیری قرار داده شده است. توجه شود که پارامتر n در این مثال مشخص نشده و به پیشفرض ۱ در نظر گرفته می شود.
مشتق جزئی:

```
1 D[f[x, y], x, y]
2 D[f[x, y], {x, 2}, {y, 3}]
```

خروجی آن در ترمینال به شکل زیر است:

```
Out[•]= f^(1,1) [x, y]
```

```
Out[•]= f^(2,3) [x, y]
```

مثال کاربردی:

```
1 D[x^3 + y^2, {x, 2}, {y, 1}] Out: 0
```

بخش سوم - انتگرال گیری

```
1 Integrate[f[x], x]
```

که خروجی آن در ترمینال به صورت زیر نمایش داده می شود:

```
Out[•]= \int f[x] dx
```

یک مثال از انتگرال نامعین:

```
1 Integrate[Sin[x], x] Out: -Cos[x]
```

انتگرال دوگانه نامعین:

```
1 Integrate[f[x, y], x, y]
```

که خروجی آن در ترمینال به صورت زیر نمایش داده می شود:

```
Out[•]= \int \int f[x, y] dy dx
```

یک مثال از انتگرال دوگانه نامعین:

```
1 Integrate[x*y + 1, x, y] Out: x y + (x^2 y^2)/4
```

انتگرال معین:

```
1 Integrate[1/(x^3 + 1), {x, 0, 1}] Out: 1/18 (2 Sqrt[3] \[Pi] + Log[64])
```

نکته: جواب انتگرال گیری‌های معین است.

یک مثال از انتگرال دوگانه معین:

```
1 Integrate[x*y + 1, {x, 0, t}, {y, -x, x}] Out: t^2
```

انتگرال‌های عددی:

```
1 NIntegrate[Sin[Sin[x]], {x, 0, 4}] Out: 1.45747
```

اما چرا از این روش استفاده می‌شود. باید از روش قبل انتگرال بالا را حساب کنیم:

```
1 Integrate[Sin[Sin[x]], {x, 0, 4}]
```

خروجی به شکل زیر خواهد بود:

$$Out[\bullet] = \int_0^4 \sin[\sin[x]] dx$$

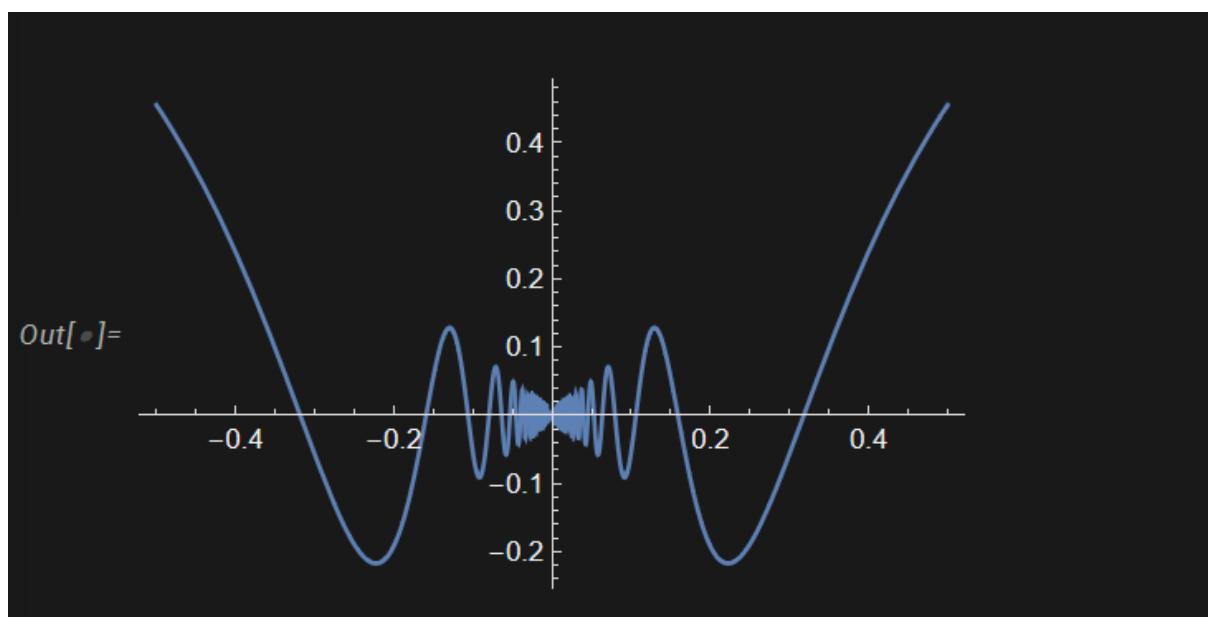
دلیل آن بر این است که این انتگرال به صورت بسته با توابع ابتدایی بیان شدنی نیست. همین است که Integrate در معمولاً جواب نمادین برنمی‌گرداند.
یک مثال دیگر:

```
1 a = 2; NIntegrate[a*Sin[Sin[x]], {x, 0, 4}] Out: 2.91494
```

جلسه پنجم: رسم توابع
۵ - ۱ رسم توابع در فضای دو بعد:

```
1 Plot[x*Sin[1/x], {x, -1/2, 1/2}]
```

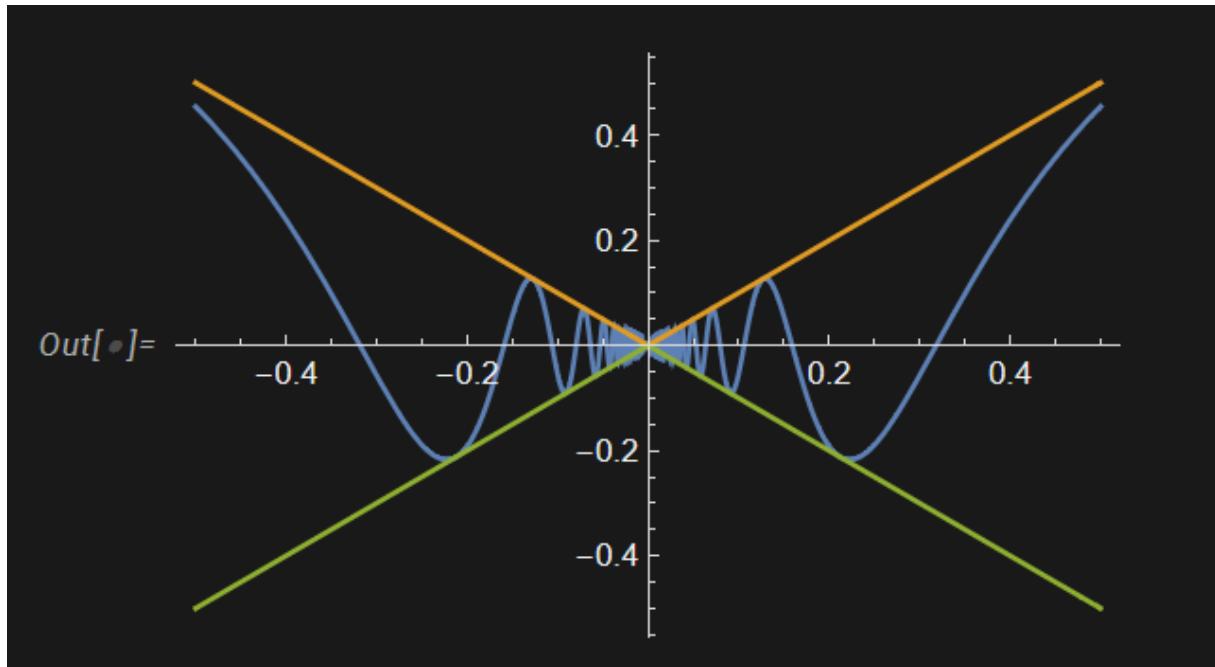
خروجی:



رسم سه تابع در یک دستگاه مختصات:

```
1 Plot[{x*Sin[1/x], Abs[x], -Abs[x]}, {x, -1/2, 1/2}]
```

خروجی:



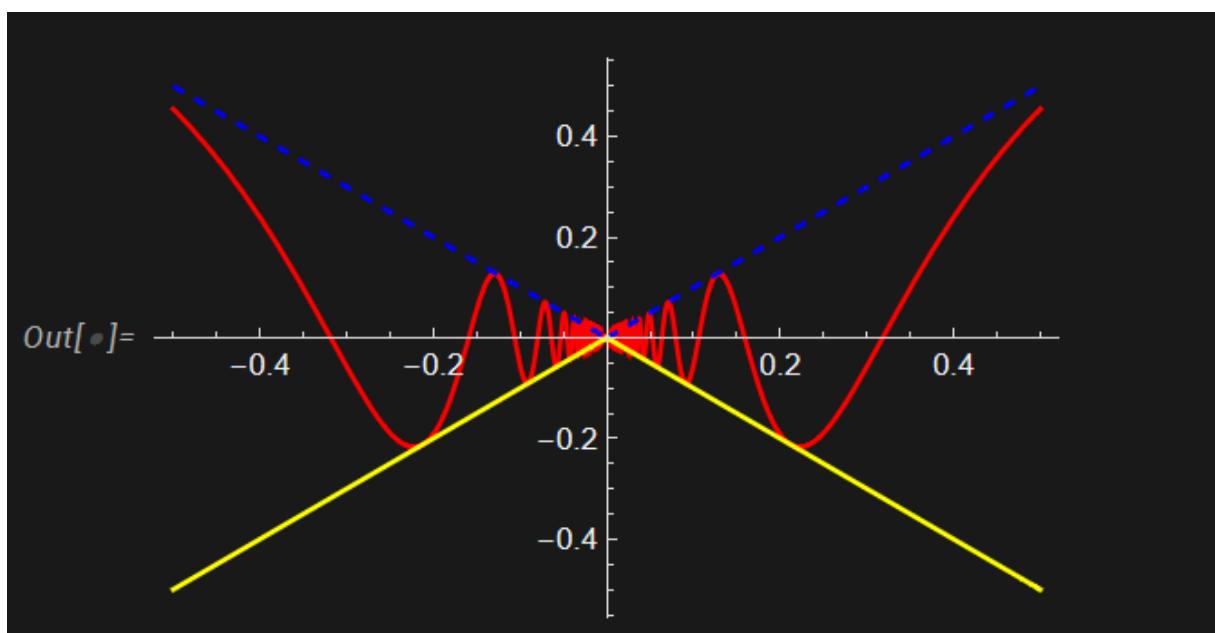
همانطور که مشاهده می شود هر سه تابع در یک دستگاه مختصات با رنگ های مختلف رسم می شود. برای تغییر رنگ و استایل توابع، در ادامه دستور قبل کد زیر نوشته می شود:

```
1 PlotStyle -> { Red, Directive[Dashed, Blue], Yellow}
```

کد کامل با استایل اضافه شده:

```
1 Plot[{x*Sin[1/x], Abs[x], -Abs[x]}, {x, -1/2, 1/2},  
2 PlotStyle -> { Red, Directive[Dashed, Blue], Yellow}]
```

خروجی:



نکته ۱ : دستور Directive برای اعمال چندین دستور برای یک نمودار استفاده می شود.

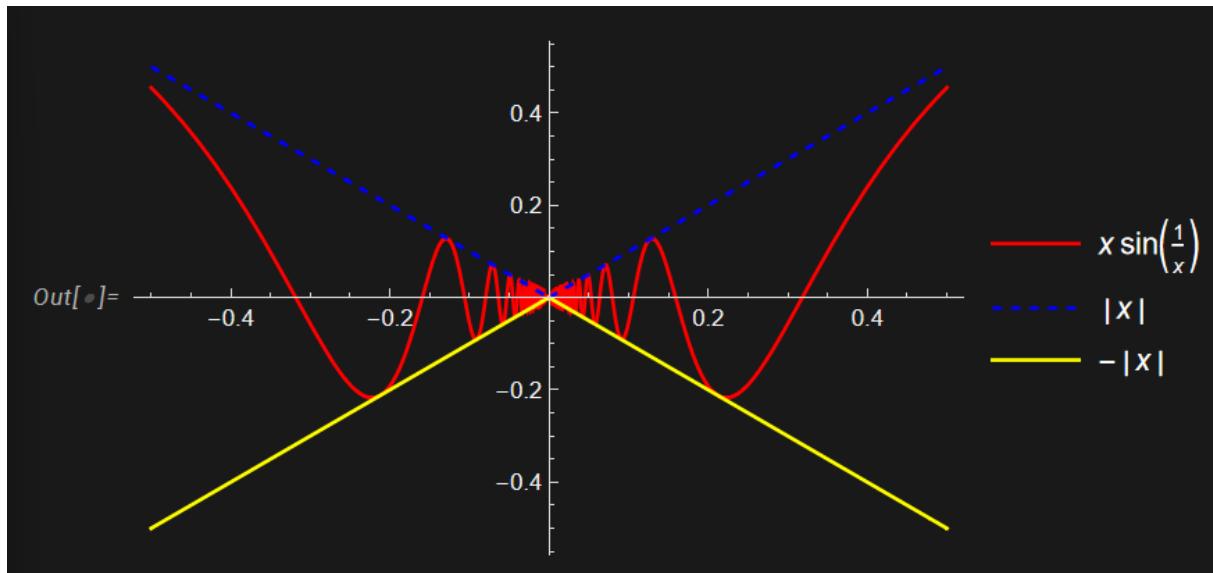
نکته ۲ : بعد از `Plot` نمی‌توانیم از ؛ استفاده کنیم، زیرا `plot` یک تابع است و ذخیره سازی آن ممکن نیست و صرفاً برای نمایش استفاده می‌شود.
برای راهنمای قرار دادن برای نمودار، بعد از تغییر رنگ در همان کروشه از دستور زیر استفاده می‌نماییم:

```
1 PlotLegends -> "Expressions"
```

*توضیحاتی درباره نمودار نمایش داده خواهد شد.
کد کامل با بخش اضافه شده:

```
1 Plot[{x*Sin[1/x], Abs[x], -Abs[x]}, {x, -1/2, 1/2},  
2 PlotStyle -> {Red, Directive[Dashed, Blue], Yellow},  
3 PlotLegends -> "Expressions"]
```

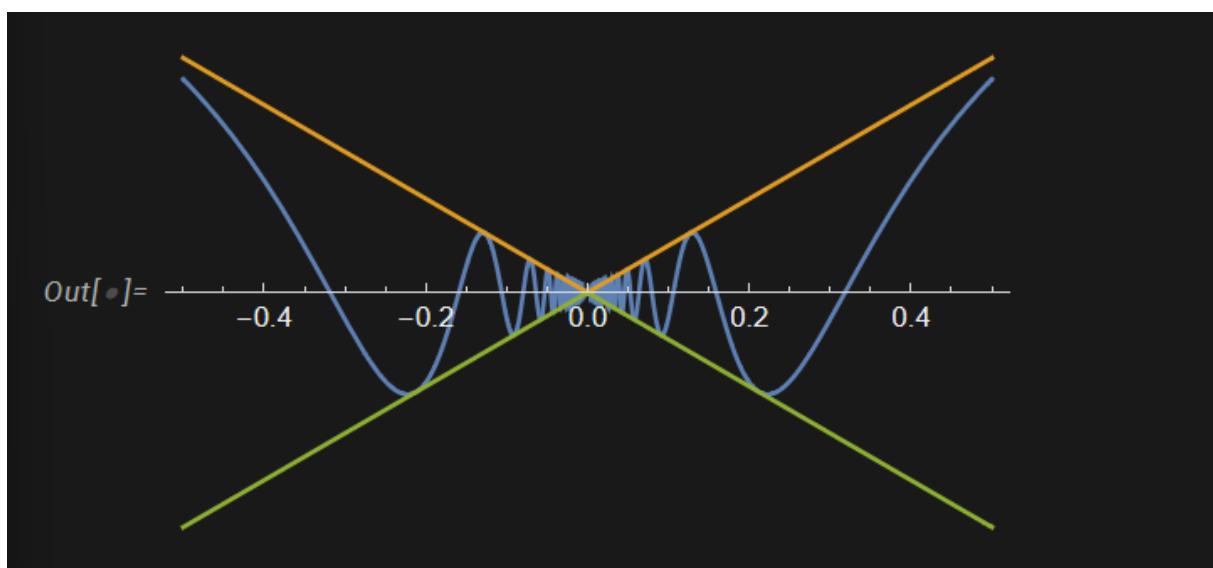
خروجی:



برای نمایش ندادن یکی از محورها یا هر دوی آنها (محورهای x و y) مانند زیر عمل می‌کنیم:

```
1 Plot[{x*Sin[1/x], Abs[x], -Abs[x]}, {x, -1/2, 1/2},  
2 Axes -> {True, False}]
```

خروجی:



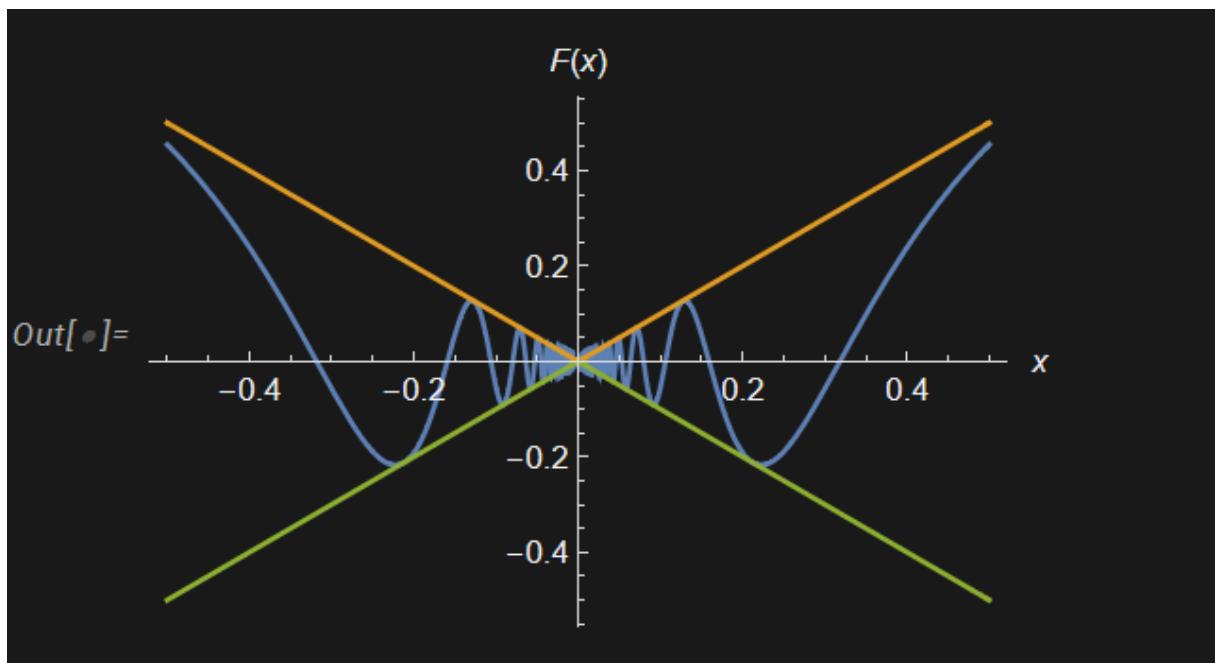
نکته: مقدار `True` به این معنی است که محور نمایش داده شود و مقدار `False` به این معنی است که محور نمایش داده نشود. همانطور که مشاهده می‌شود در این مثال محور x نمایش داده شده و محور y حذف شده است. اگر مقدار x هم قرار دهیم، محور x نیز نمایش داده نمی‌شود.
برای نام‌گذاری برای محورها از دستور زیر استفاده می‌کنیم:

```
1 AxesLabel -> {x, F[x]}
```

کد کامل با بخش اضافه شده:

```
1 Plot[{x*Sin[1/x], Abs[x], -Abs[x]}, {x, -1/2, 1/2},  
2 AxesLabel -> {x, F[x]}]
```

خروجی:



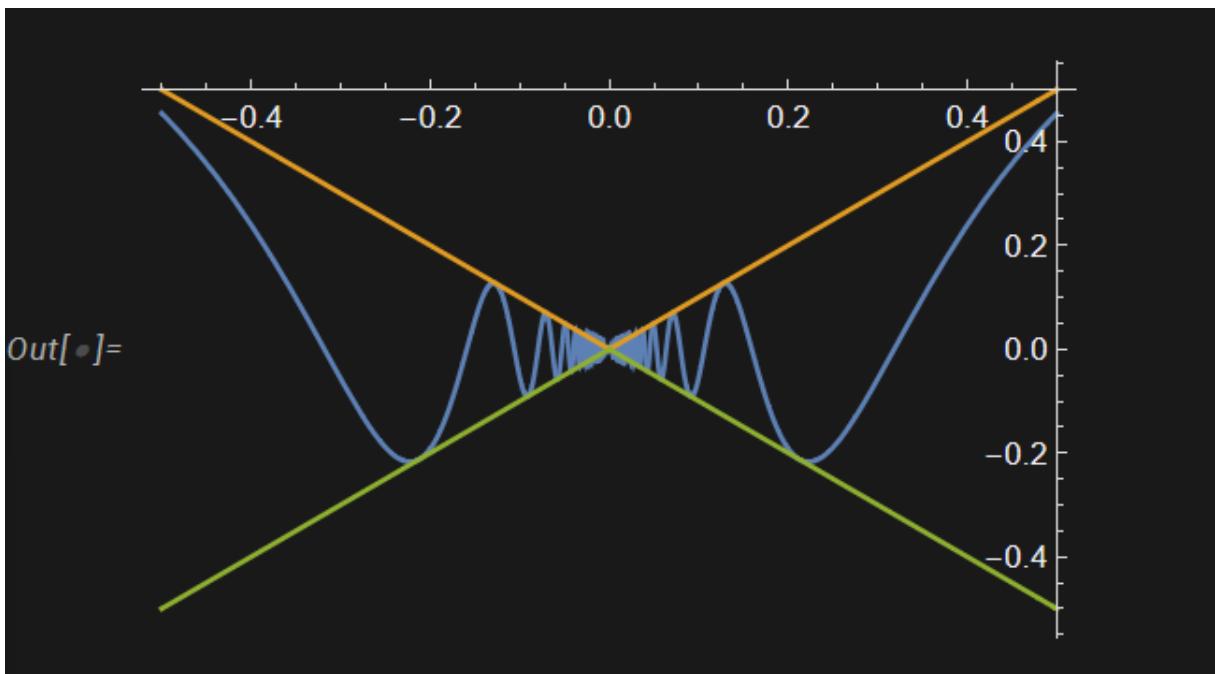
نکته: در این مثال، نام محور y به محور $F(x)$ تغییر داده شده.
برای جایگزینی نمودار از دستور زیر استفاده می‌کنیم:

```
1 AxesOrigin -> {.5, .5}
```

این کد به این معنی است که نمودار ۵.۰ واحد به سمت چپ و راست و ۵.۰ واحد به سمت پایین و بالا جایه‌جا شود.
کد کامل با بخش اضافه شده:

```
1 Plot[{x*Sin[1/x], Abs[x], -Abs[x]}, {x, -1/2, 1/2},  
2 AxesOrigin -> {.5, .5}]
```

خروجی:



Out[•]=

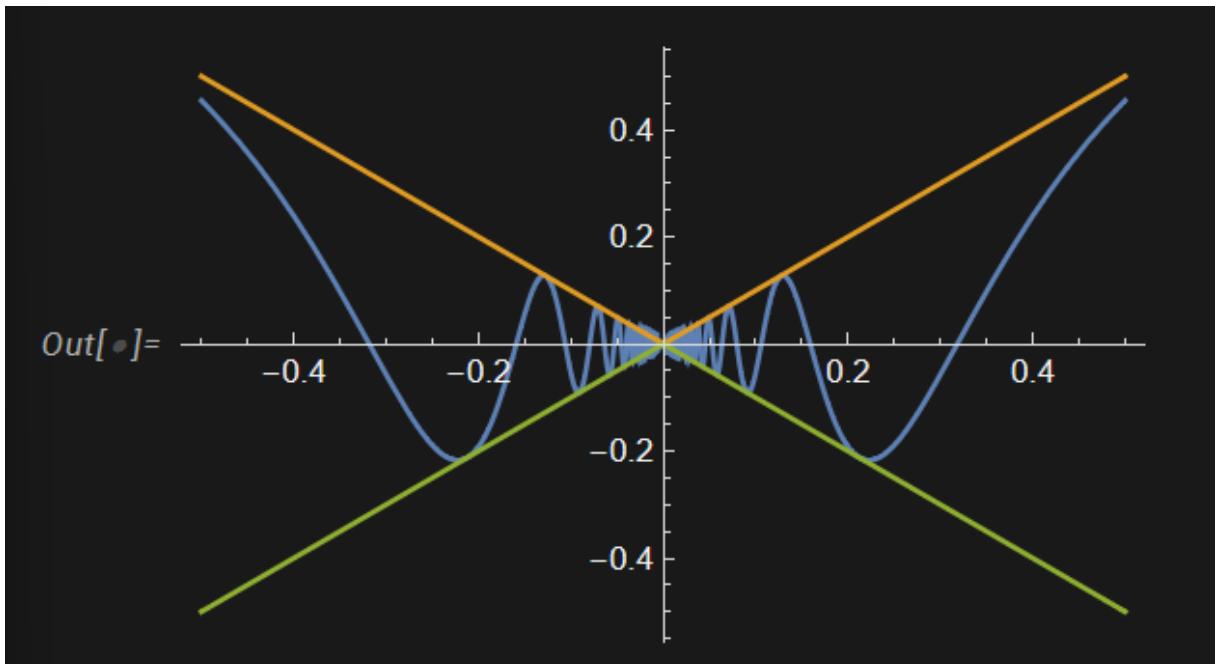
اگر نمودار کامل نمایش داده نشود، می‌توانیم از دستور زیر برای نمایش کامل آن استفاده کنیم:

```
1 PlotRange -> Full
2 //or
3 PlotRange -> All
```

کد کامل با بخش اضافه شده:

```
1 Plot[{x*Sin[1/x], Abs[x], -Abs[x]}, {x, -1/2, 1/2}, PlotRange -> Full]
```

خروجی:



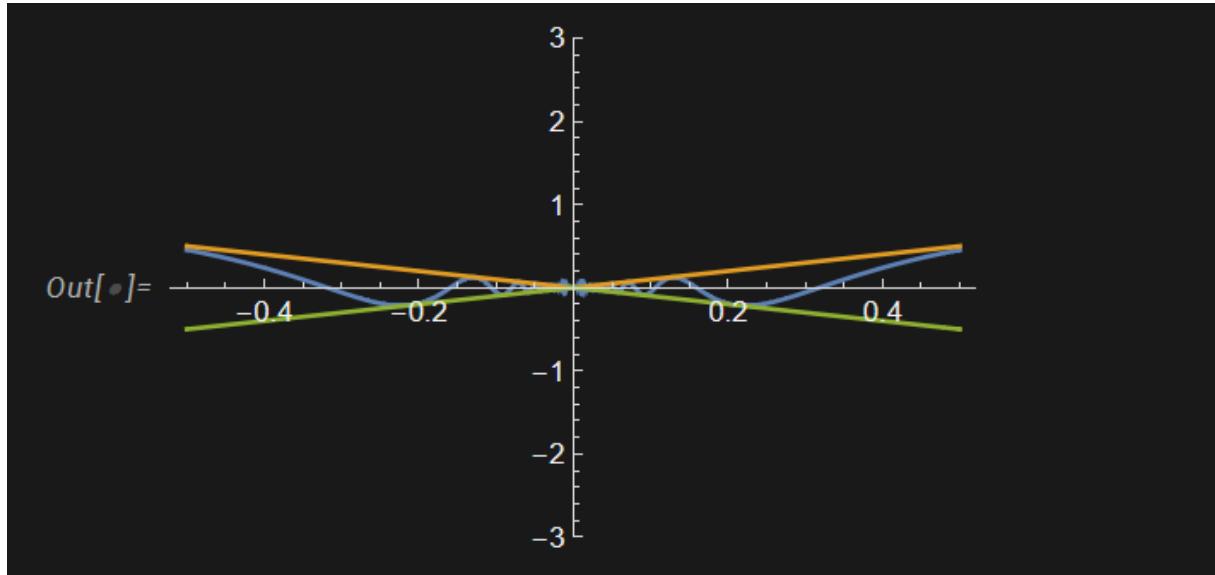
و اگر بخواهیم تا حدی مشخص نمایش داده شود، به جای Full و All آن عدد مشخص را تایپ می‌کنیم:

```
1 PlotRange -> 3
```

کد کامل با بخش اضافه شده:

```
1 Plot[{x*Sin[1/x], Abs[x], -Abs[x]}, {x, -1/2, 1/2}, PlotRange -> 3]
```

خروجی:



برای استفاده از رنگ رنگین کمانی در نمودارهای از دستور زیر استفاده می‌کنیم:

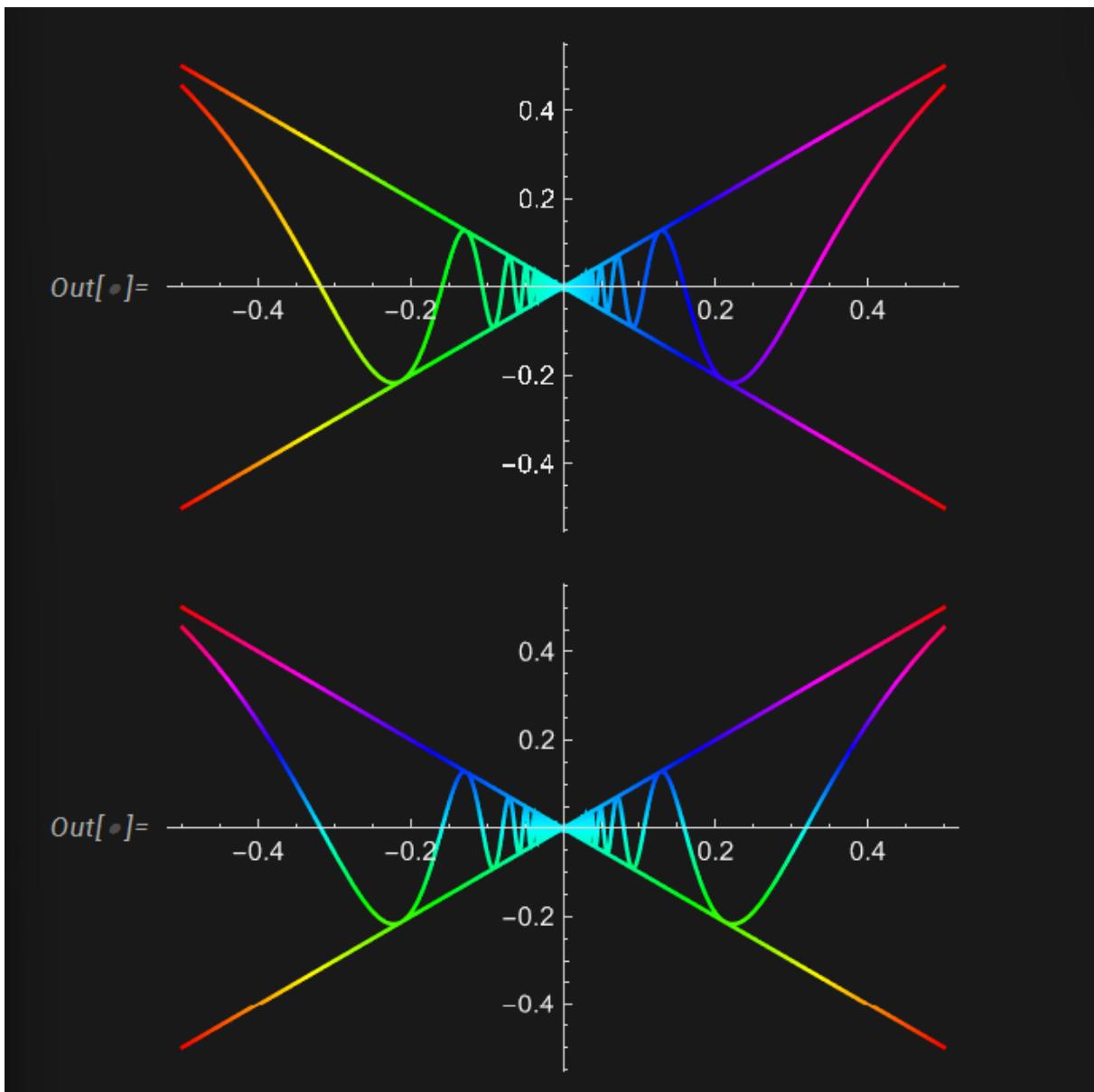
```
1 ColorFunction -> Function[{x, y}, Hue[y]]
```

نکته: $Hue[x]$ به این معنی است که جهت تغییر رنگ به سمت محور x است. اگر به جای x مقدار y را قرار دهیم. جهت تغییر رنگ‌ها به سمت محور y خواهد بود. کد کامل با بخش اضافه شده:

```
1 Plot[{x*Sin[1/x], Abs[x], -Abs[x]}, {x, -1/2, 1/2},
2   ColorFunction -> Function[{x, y}, Hue[x]]]
```

```
3 Plot[{x*Sin[1/x], Abs[x], -Abs[x]}, {x, -1/2, 1/2},
4   ColorFunction -> Function[{x, y}, Hue[y]]]
```

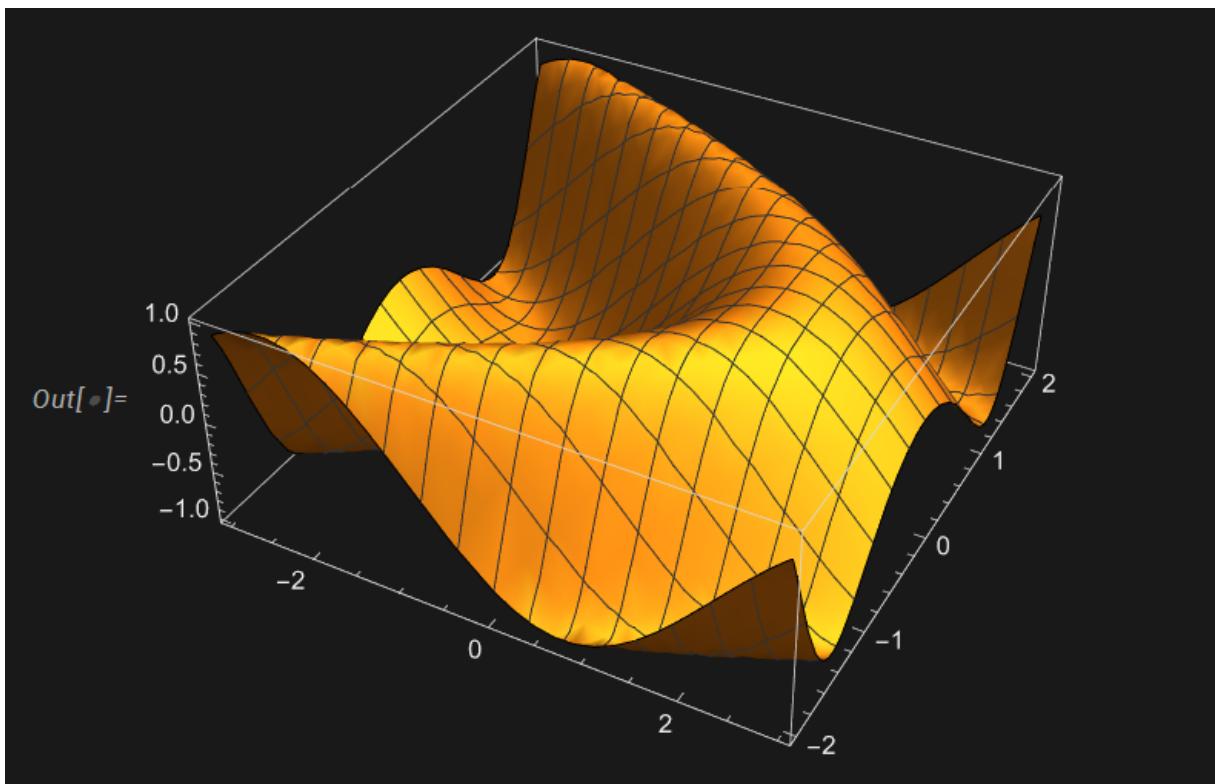
خروجی:



نکته: اگر از ColorFunction استفاده کنیم، نمی‌توانیم در دستور از Plot Legends را نیز استفاده کنیم.
۵ - ۲ - رسم رویه‌ها
سه بعدی:

```
Plot3D[Sin[x + y^2], {x, -3, 3}, {y, -2, 2}]
```

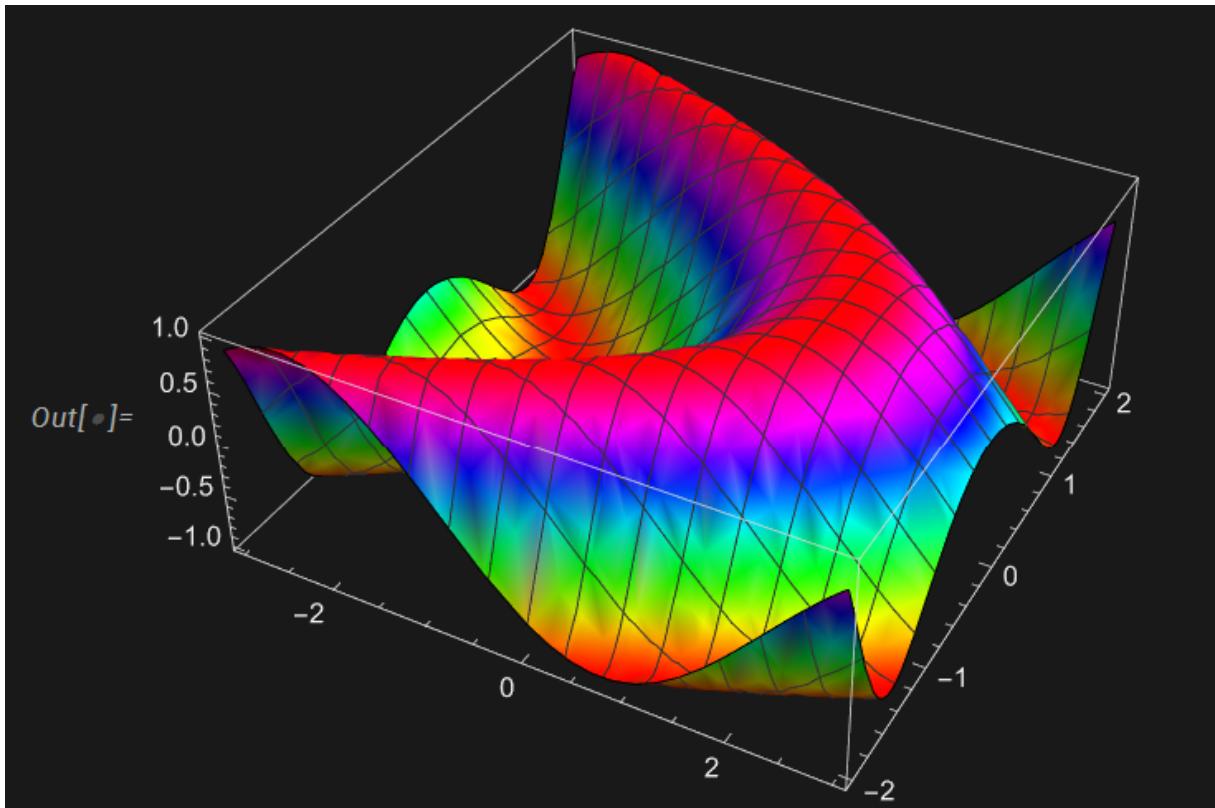
خروجی:



از **ColorFinder** می‌توانیم در روش‌های سه بعدی هم استفاده کنیم و بهتر است از $\text{Hue}[z]$ در این دستورها استفاده کنیم.

```
Plot3D[Sin[x + y^2], {x, -3, 3}, {y, -2, 2} ,
 ColorFunction -> Function[{x, y, z}, Hue[z]]]
```

خروجی:



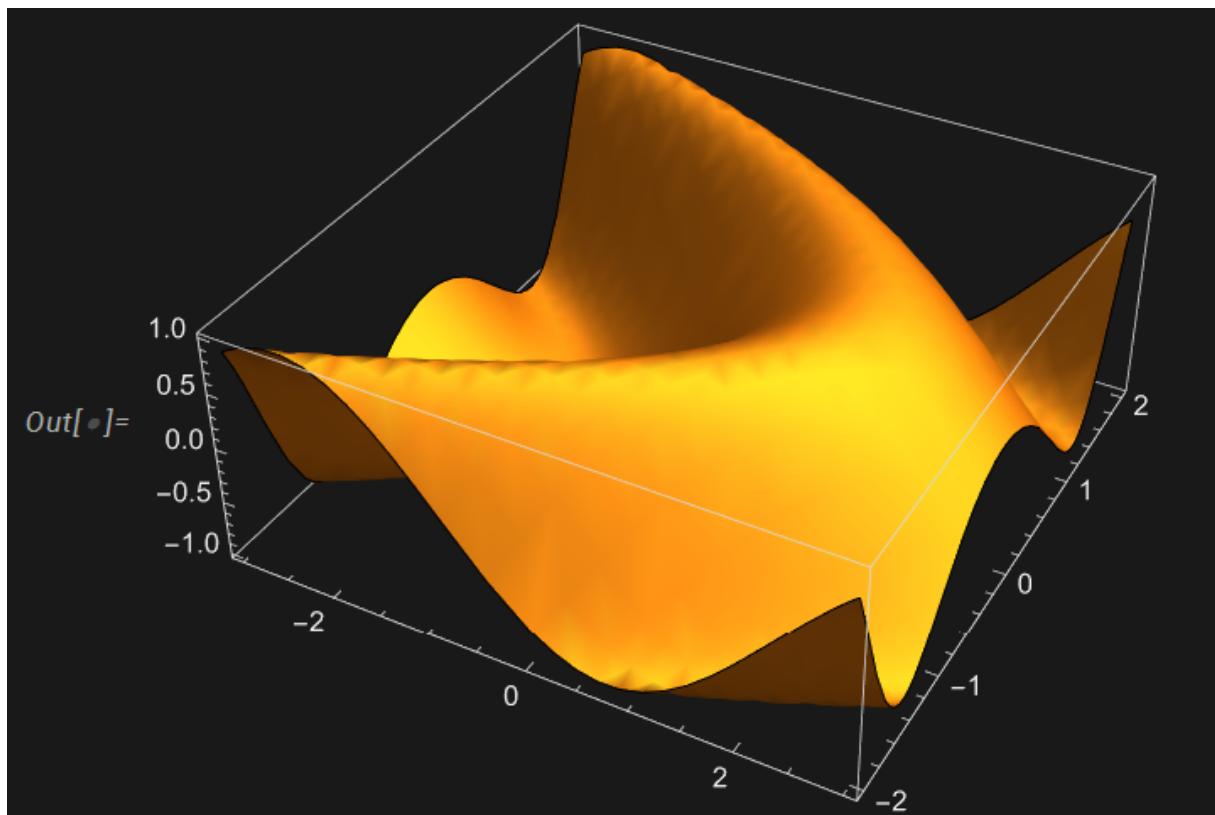
برای حذف چهارخانه‌های روی رویه از دستور زیر استفاده می‌کنیم:

```
1 Mesh -> None
```

کد کامل با بخش اضافه شده:

```
1 Plot3D[Sin[x + y^2], {x, -3, 3}, {y, -2, 2}, Mesh -> None]
```

خروجی:

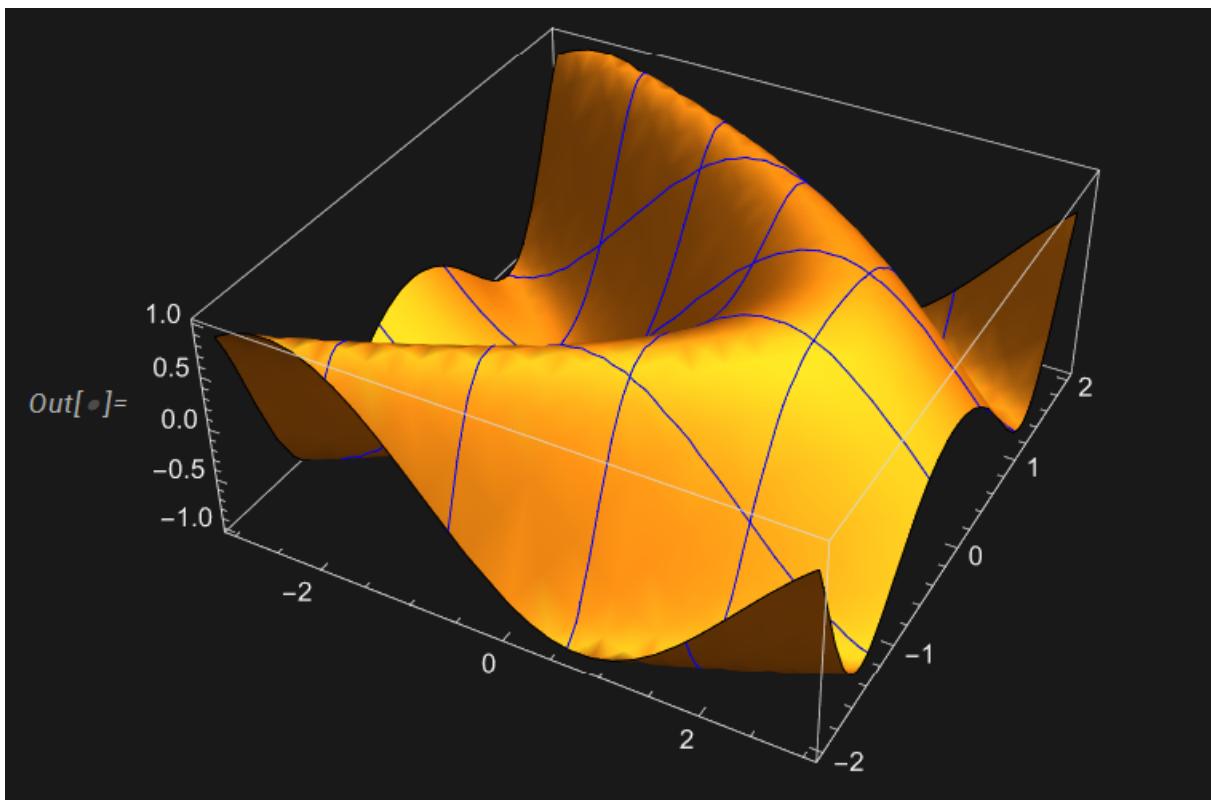


اگر به جای none از یک عدد استفاده کنیم، به تعداد آن عدد mesh در سمت x و y کشیده می‌شود:

```
1 Plot3D[Sin[x + y^2], {x, -3, 3}, {y, -2, 2}, Mesh -> 4,  
2 MeshStyle -> Blue]
```

نکته: در این کد رنگ Mesh هستش.

خروجی:



٣ - رسم توابع پارامتری توابع دو بعدی: ٥

```

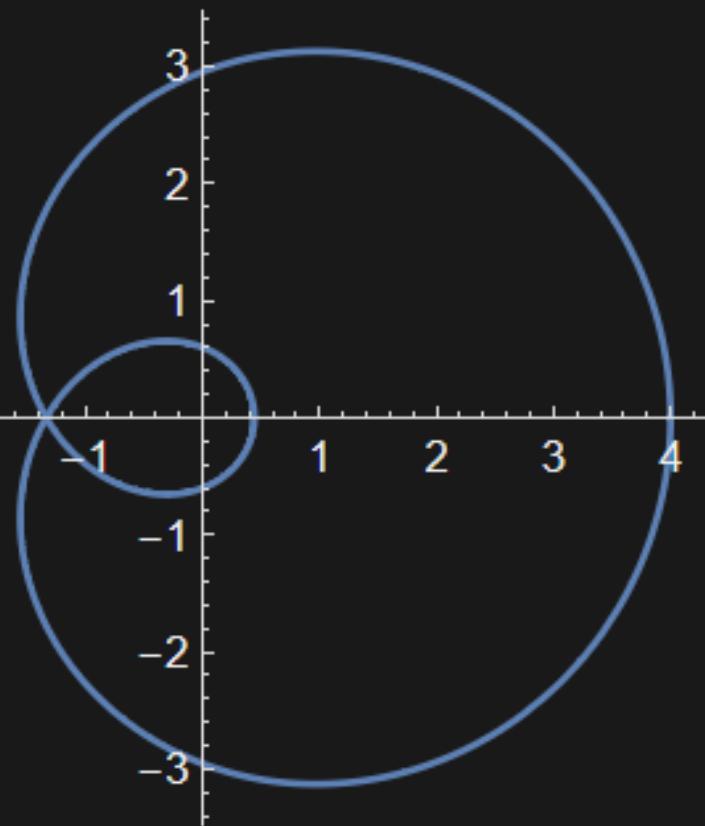
1 h = 1/(s - 1/2)^2 /. s -> Exp[I w]
2 ParametricPlot[{Re[h], Im[h]}, {w, 0, 2Pi}]

```

خروجی:

$$Out[•]= \frac{1}{\left(-\frac{1}{2} + e^{\pm w}\right)^2}$$

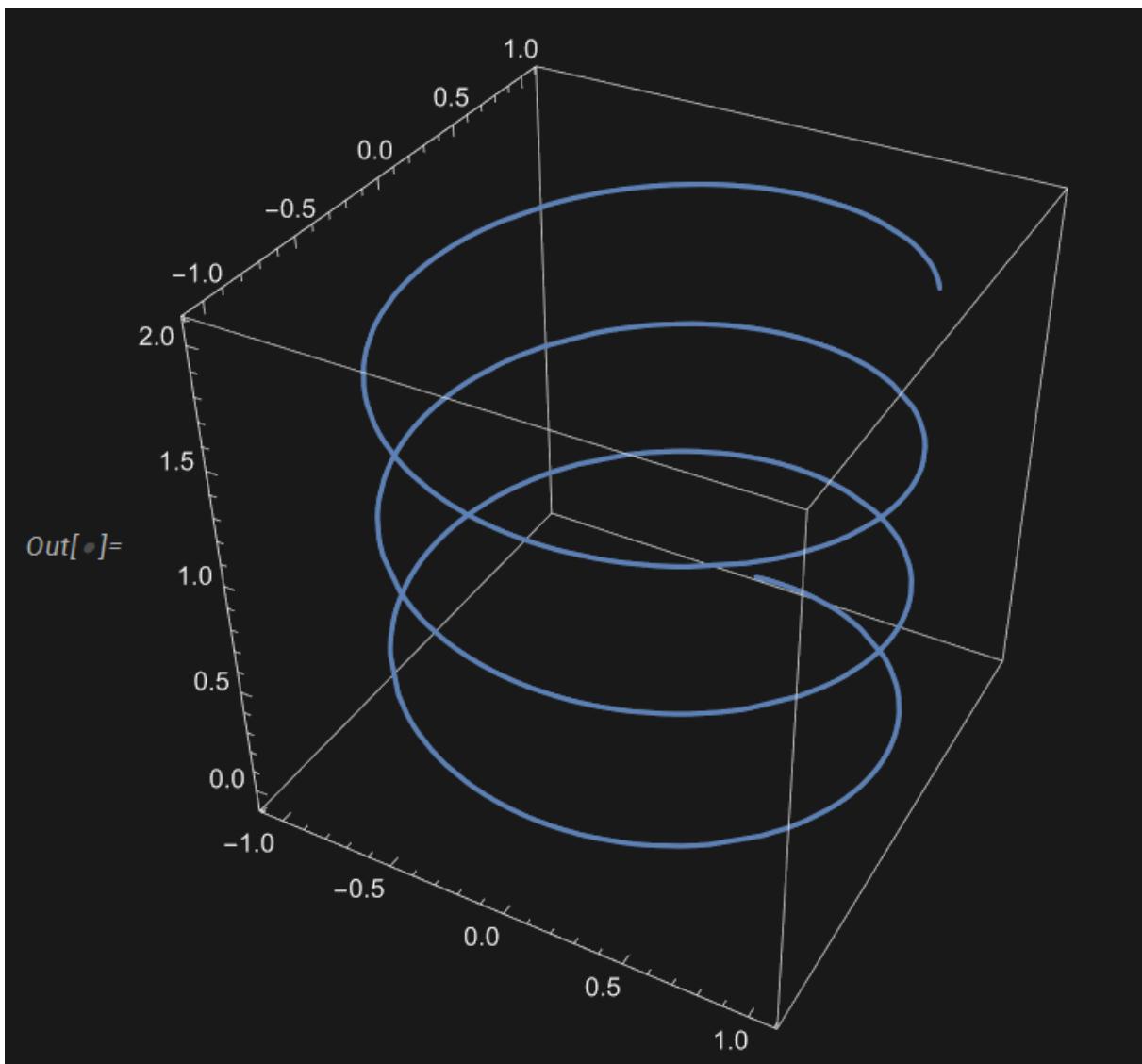
$$Out[•]=$$



توابع سه بعدی:

```
ParametricPlot3D[{Sin[x], Cos[x], x/10}, {x, 0, 20}]
```

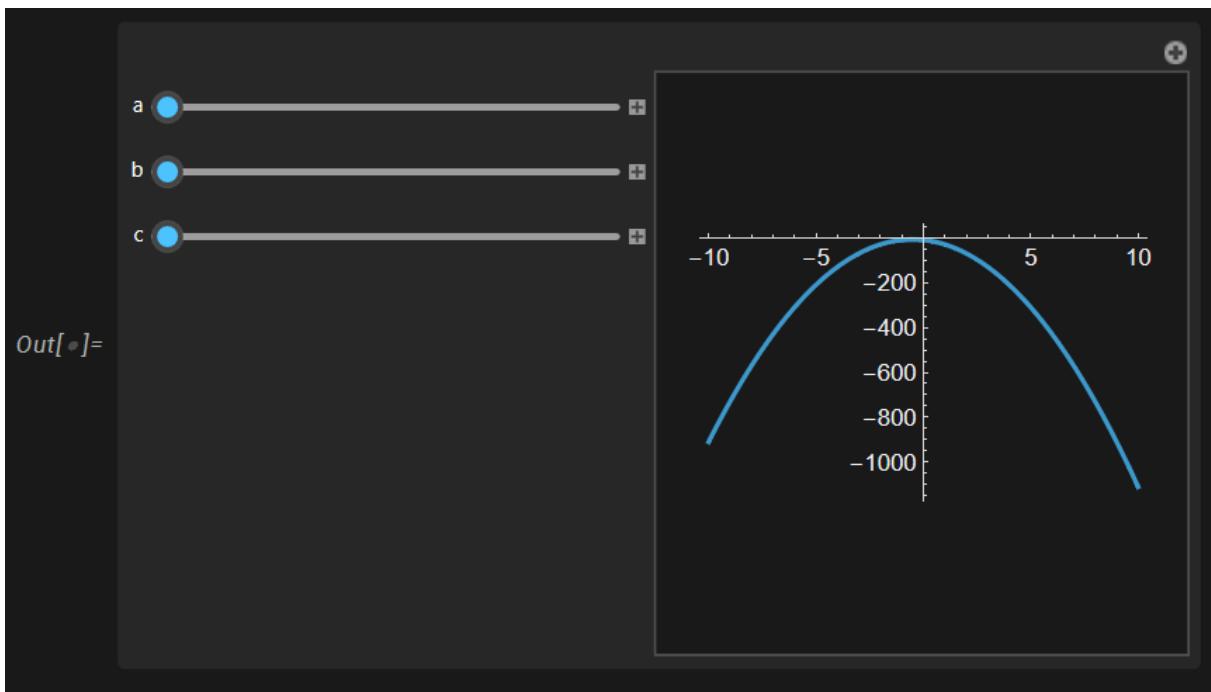
خروجی:



۴ - رسم توابع با پارامترهای نامعلوم

```
Manipulate[Plot[a*x^2 + b*x + c, {x, -10, 10}], {a, -10, 10},
           {b, -10, 10}, {c, -10, 10}]
```

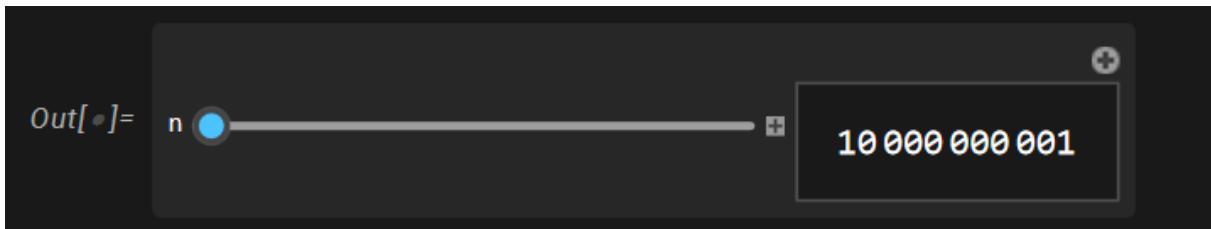
خروجی:



از Manipulate نیز می‌توان برای فاکتورگیری و چندجمله‌ای‌ها نیز استفاده کرد:

```
Manipulate[Factor[n^n + 1], {n, 10, 100, 13}]
```

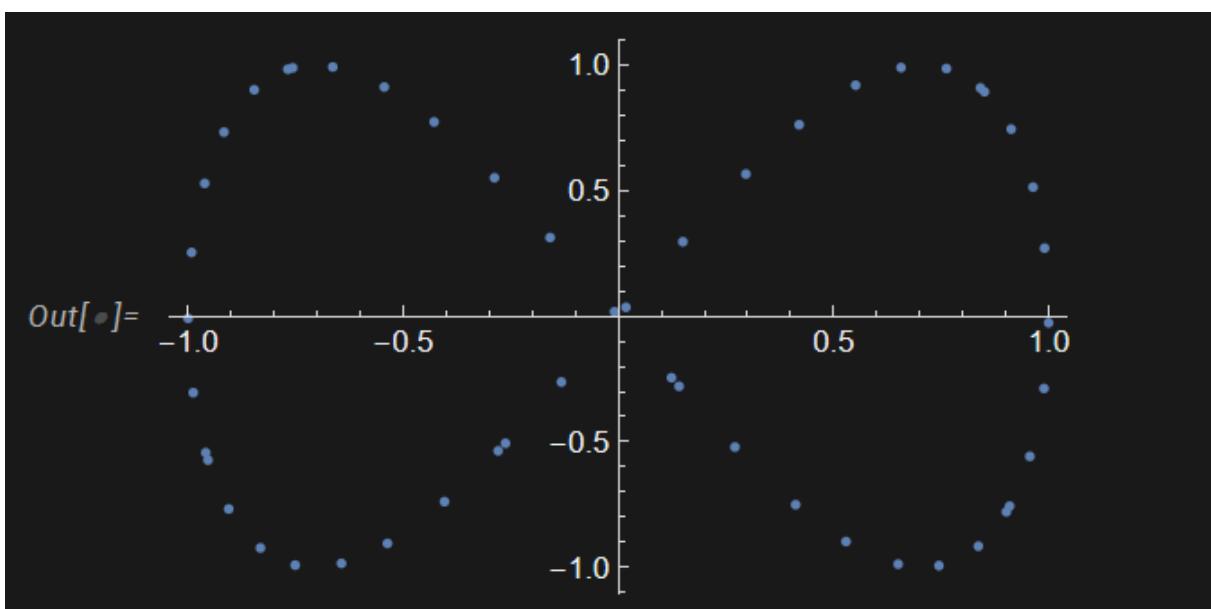
خروجی:



۵- رسم توابع با استفاده از نقاط

```
ListPlot[Table[{Sin[x], Sin[2 x]}, {x, 50}]]
```

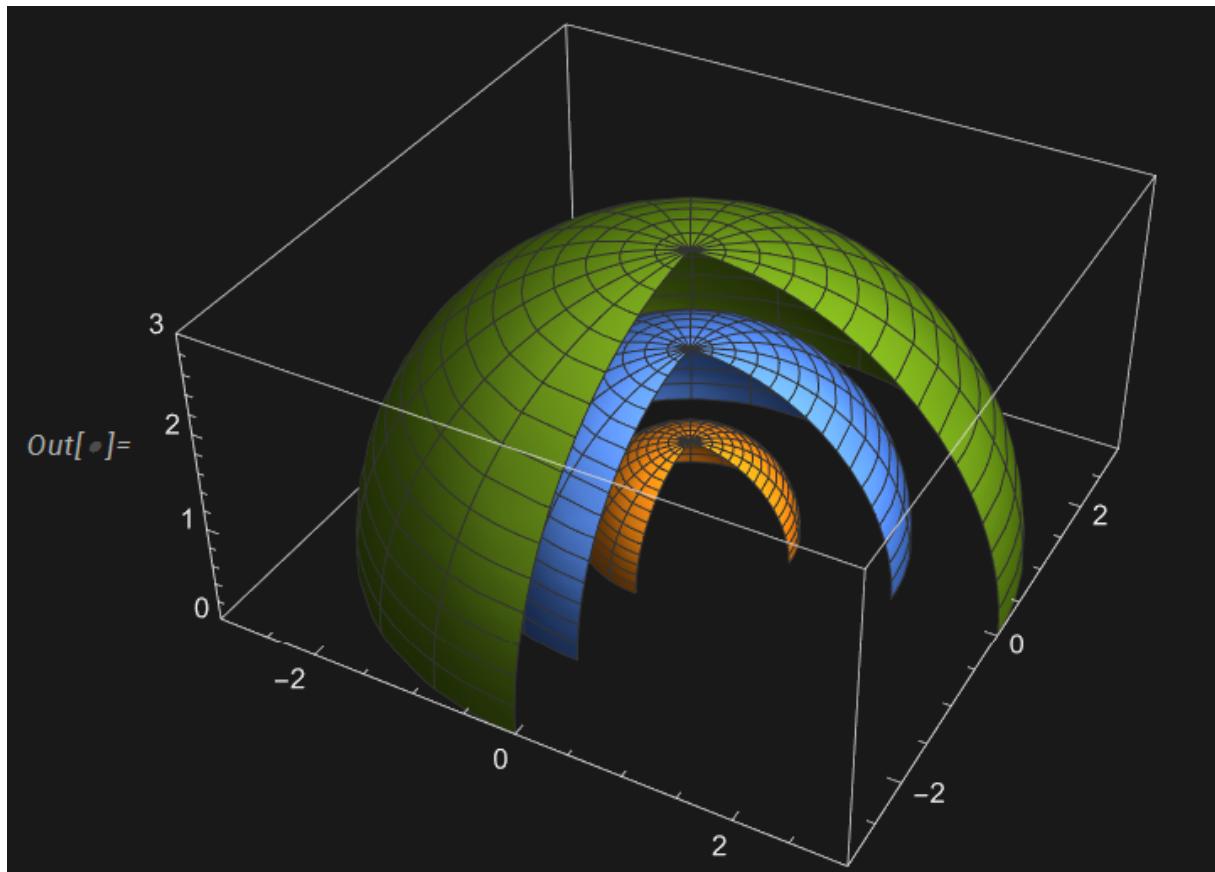
خروجی:



۶ - رسم منحنی در مختصات‌های مختلف مختصات کروی: مثال اول

```
1 SphericalPlot3D[{1, 2, 3}, {\[Theta], 0, Pi/2}, {\[CapitalPhi], 0,
2 3*Pi/2}]
```

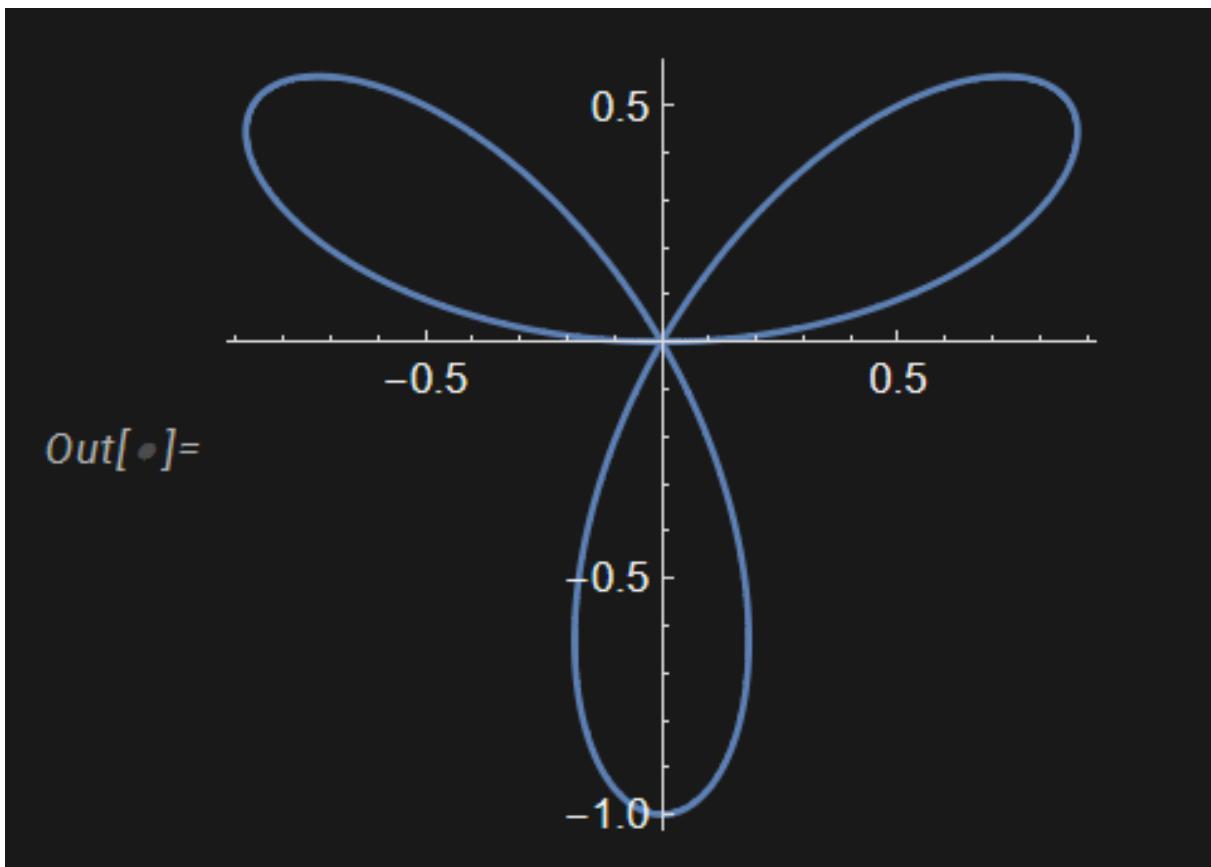
خروجی:



مختصات کروی: مثال دوم

```
1 PolarPlot[Sin[3*t], {t, 0, Pi}]
```

خروجی:



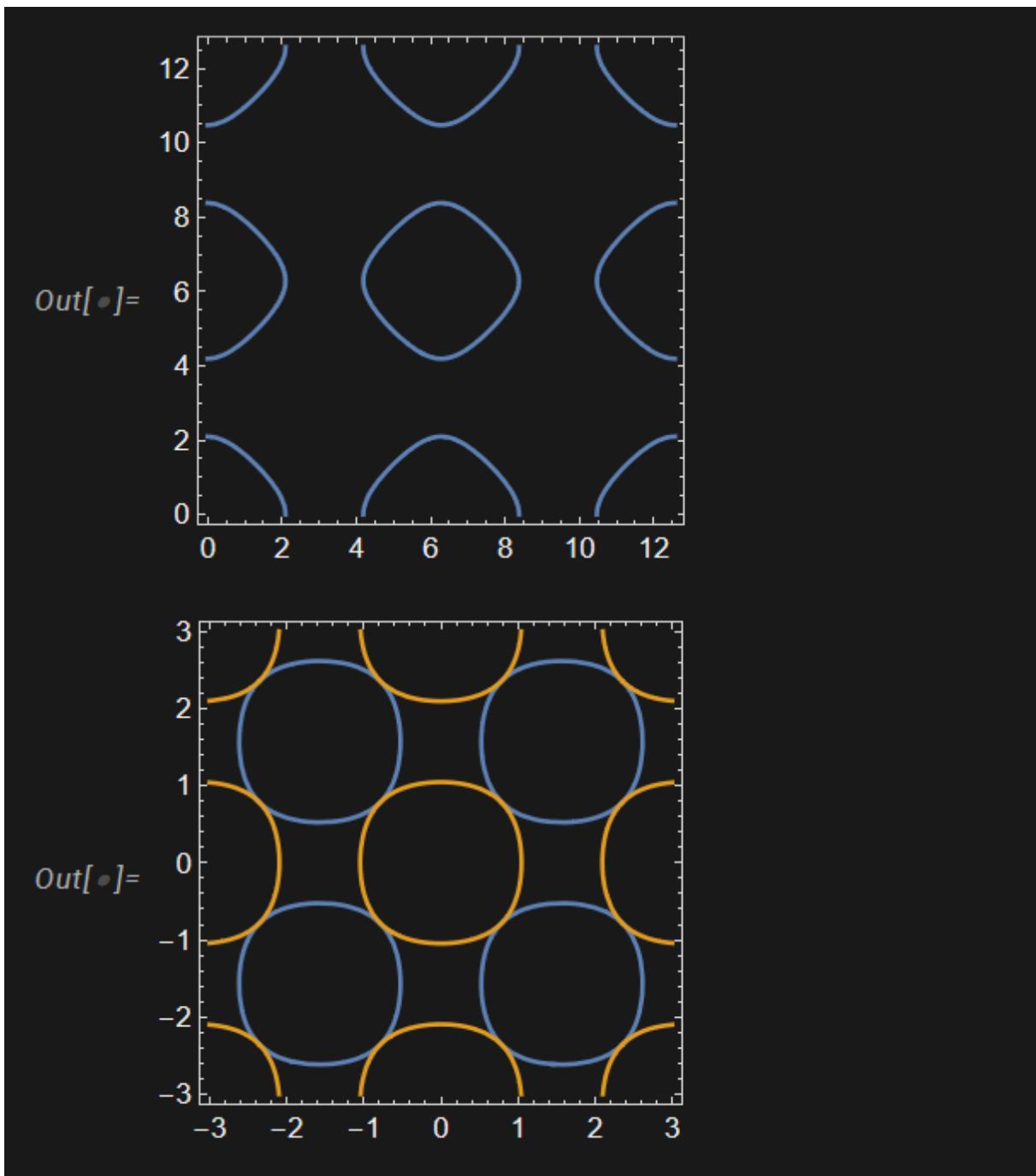
Out[•]=

۷ - رسم محدوده جواب نامعادلات جبری

```

1 ContourPlot[Cos[x] + Cos[y] == 1/2, {x, 0, 4*Pi}, {y, 0, 4*Pi}]
2 ContourPlot[{Abs[Sin[x]* Sin[y]] == .5 ,
3   Abs[Cos[x]*Cos[y]] == .5}, {x, -3, 3}, {y, -3, 3}]
```

خروجی:



جلسه ششم: حل معادلات ۶ - ۱ حل معادلات جبری
مثال ۱ :

```
1 Solve[a*x^2 + b*x + c == 0, x] Out: {{x -> (-b - Sqrt[b^2 - 4 a c])/(2 a)}, {x -> (-b + Sqrt[b^2 - 4 a c])/(2 a)}}
```

* ریشه‌های معادله درجه دو را بدست می‌آورد.
مثال ۲ :

```
1 Solve[x^2 + y^2 == 2 && x - y == 1 , {x, y}] Out: {{x -> 1/2 (1 - Sqrt[3]), y -> 1/2 (-1 - Sqrt[3])}, {x -> 1/2 (1 + Sqrt[3]), y -> 1/2 (-1 + Sqrt[3])}}
```

* این معادله دو نقطه و دو جواب دارد که آن نقطه تقاطع این دو معادله است.
برای اینکه جوابها را جداگانه بدهد:

```
| sol = Solve[x^2 + y^2 == 2 && x - y == 1, {x, y}]  
| x /. sol  
Out: {1/2 (1 - Sqrt[3]), 1/2 (1 + Sqrt[3])}
```

* مثال ۳ : **x** های جواب معدله را نشان می دهد.

```
Solve[x == 1 && x == 2, x] Out: {}
```

* این دسته معادله‌ها جواب ندارند. علامت یعنی دستگاه جوابی ندارد.
مثال ۴ :

```
Solve[x == x, x] Out: {{}}
```

* در این مثال مجمع حواب بی‌نهایت است. علامت یعنی دستگاه بی‌نهایت جواب دارد.
مثال ۵ :

```
Solve[(x^4 - 1)*(x - 4) == 0, x, Reals] Out: {{x -> -1}, {x -> 1}, {x -> 4}}
```

* ریشه‌های Real هستند در خروجی نمایش داده خواهد شد.
اگر یک معادله با دستور Solve حل نشود، ممکن است با استفاده از دستور NSolve حل بشود.

```
NSolve[(x - 1)*(x^4 - 4) == 0, Reals] Out: {{x -> -1.41421}, {x -> 1.}, {x -> 1.41421}}
```

۶- یافتن ریشه معادلات غیرخطی به روش نیوتون و سکانت
روش نیوتون (یک نقطه ورودی دارد)

```
FindRoot[Sin[x] + Exp[x], {x, 0}] Out: {x -> -0.588533}
```

روش سکانت (دو نقطه ورودی دارد)

```
FindRoot[Sin[x] + Exp[x], {x, 0, 1}]           Out: {x -> -0.588533}
```

اگر بخواهیم از یک نقطه شروع کنیم و در یک بازه خاص دنیا جواب پگردد از روش زیر استفاده می‌کنیم (روش نیوتون)

* در این مثال، پارامتر دوم که در اینجا σ قرار داده شده نقطه شروع، دو پارامتر بعدی بازه را مشخص می‌کند. در اینجا بازه $[1, \infty)$ است.

برای بالا بردن دقیقیت محاسبات می‌توانیم از WorkingPrecision استفاده کنیم. به مثال زیر دقت کنید:

اگر بخواهیم دقت پسیار پسیار زیاد شود:

```
Needs["FunctionApproximations`"]
InterpolateRoot[Exp[x] == 2, {x, 0, 1}]      Out: {x -> 0.693147180559945309417232}
```

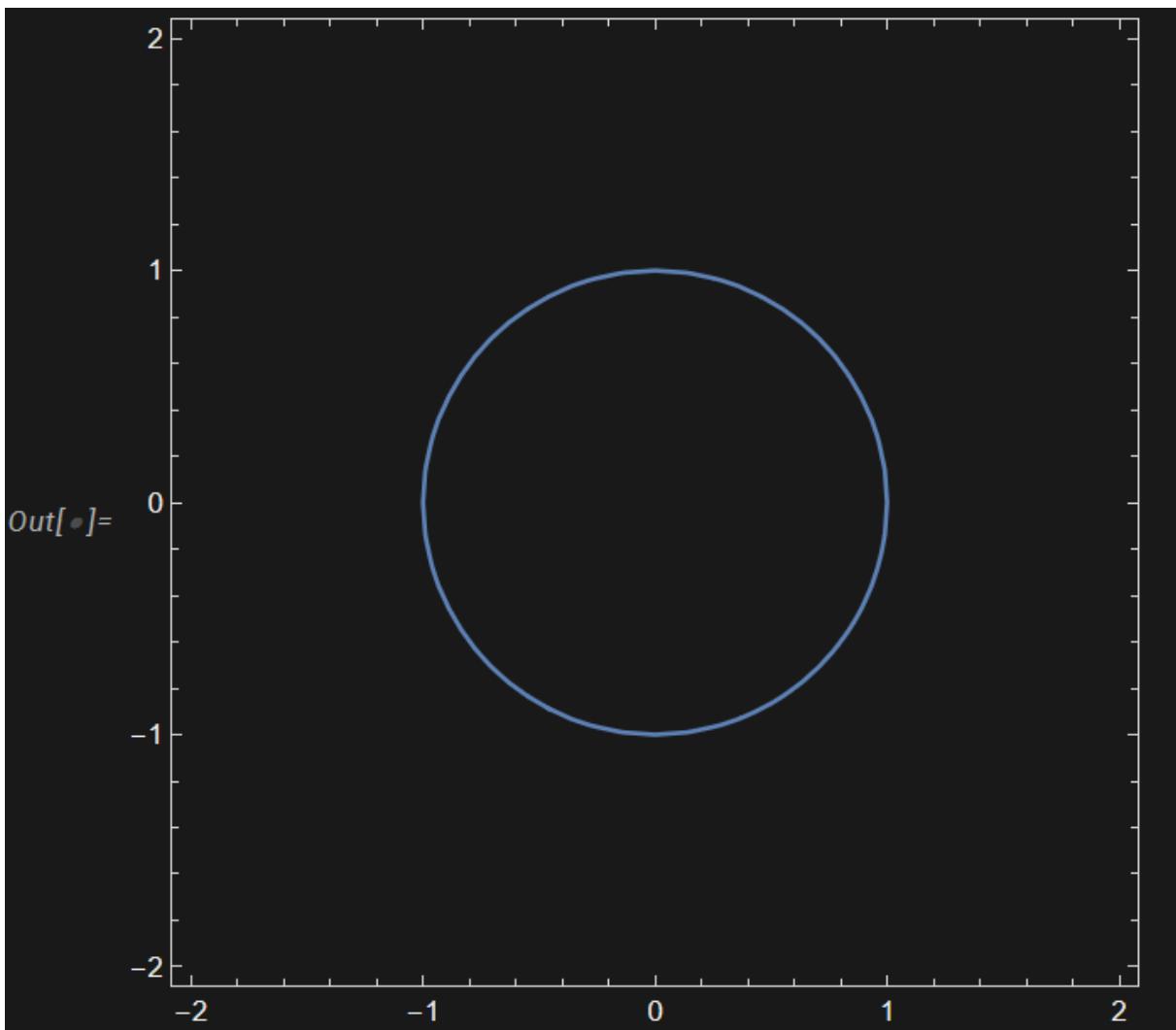
حل نامعادله:

```
Reduce[x^2 + y^2 < 1, {x, y}]  Out: -1 < x < 1 && -Sqrt[1 - x^2] < y < Sqrt[1 - x^2]
```

رسم دائرة:

```
ContourPlot[x^2 + y^2 == 1, {x, -2, 2}, {y, -2, 2}]
```

خروجی:



Out[•]=

۶ - ۳ معادلات دیفرانسیل معمولی و جزئی

حل معادله دیفرانسیل $\begin{cases} \text{روش تحلیلی (دقیق)} \\ \text{روش عددی (تقریبی)} \end{cases}$

معادلاتی که می‌توان به روش تحلیلی حل کرد:

```
DSolve[{y'[x] + y[x] == a Sin[x], y, x] Out: {{y -> Function[{x}, E^-x C[1] + 1/2 a (-Cos[x] + Sin[x])]}}
```

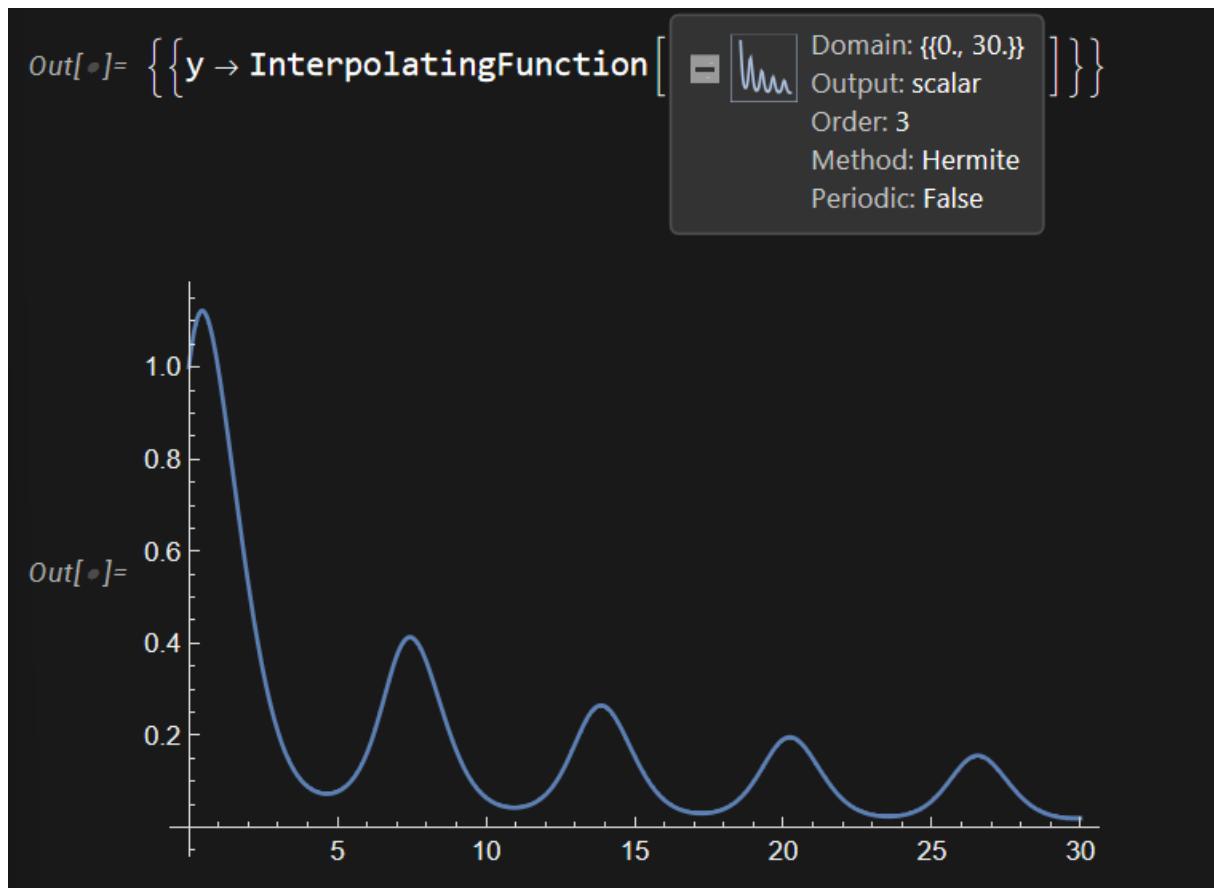
اگر بخواهیم مقدار [C] محاسبه شود، کد را اینگونه می‌نویسیم:

```
DSolve[{y'[x] + y[x] == a Sin[x], y[0] == 0, y, x] Out: {{y -> Function[{x}, -(1/2) a E^-x (-1 + E^x Cos[x] - E^x Sin[x])]}}
```

* عبارت $[y' == 0]$ در کد بالا شرط اولیه است.
معادلاتی که می‌توان به روش عددی (تقریبی) حل کرد:

```
s = NDSolve[{y'[x] == y[x]*Cos[x + y[x]], y[0] == 1}, y, {x, 0, 30}]  
Plot[Evaluate[y[x] /. s], {x, 0, 30}, PlotRange -> All]
```

خروجی:



حل معادلات پارشال:

```
1 eq = D[\[Psi][x, t], {x, 2}] == -1/\[Alpha]^2 D[\[Psi][x, t], {t, 2}]
2 DSolve[eq, \[Psi][x, t], {x, t}]
```

خروجی:

$$\text{Out}[•]= \psi^{(2,0)}[x, t] == -\frac{\psi^{(0,2)}[x, t]}{\alpha^2}$$

$$\text{Out}[•]= \left\{ \left\{ \psi[x, t] \rightarrow c_1 \left[t + \frac{x \sqrt{-\alpha^2}}{\alpha^2} \right] + c_2 \left[t - \frac{x \sqrt{-\alpha^2}}{\alpha^2} \right] \right\} \right\}$$

جلسه هفتم: بردار و ماتریس
نحوه ساخت یک وکتور (بردار) در متماتیکا:

```
1 v = {1, 2, 3} Out: {1, 2, 3}
```

نحوه ساخت یک ماتریس:

```
1 y = {{1, 2}, {3, 4}} Out: {{1, 2}, {3, 4}} // 2×2 matrix
```

اگر بخواهیم به شکل ماتریس نمایش داده شود:

```
1 y = {{1, 2}, {3, 4}} // MatrixForm
```

خروجی:

Out[•]//MatrixForm=

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$$

: Table روش ساخت یک بردار و ماتریش با دستور

```
1 Table[i^2, {i, 5}]      Out: {1, 4, 9, 16, 25}
```

اگر ماتریش شما از قانون خاصی پیروی نمی‌کند (یک ماتریس خاص است) باید خود ماتریس در Table نوشت. مثالی دیگر:

```
1 Table[i^2, {i, 5, 10}]          Out: {25, 36, 49, 64, 81, 100}
2 Table[i^2, {i, 5, 10, 2}]       Out: {25, 49, 81}
```

```
1 Table[i^2, {i, 5}, {j, 5}] // MatrixForm
2 Table[j^3, {i, 5}, {j, 5}] // MatrixForm
```

خروجی:

Out[•]//MatrixForm=

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 4 & 4 & 4 & 4 & 4 \\ 9 & 9 & 9 & 9 & 9 \\ 16 & 16 & 16 & 16 & 16 \\ 25 & 25 & 25 & 25 & 25 \end{pmatrix}$$

Out[•]//MatrixForm=

$$\begin{pmatrix} 1 & 8 & 27 & 64 & 125 \\ 1 & 8 & 27 & 64 & 125 \\ 1 & 8 & 27 & 64 & 125 \\ 1 & 8 & 27 & 64 & 125 \\ 1 & 8 & 27 & 64 & 125 \end{pmatrix}$$

* نجوه و ترتیب قرارگیری i و j در Table بسیار مهم است.
۷ - ۱ اعمال اصلی و گرفتن درایه از ماتریس

```
1 M = Table[j^2, {i, 5}, {j, 5}] // MatrixForm
2 M + M // MatrixForm;           // Summation
3 M . M // MatrixForm;           // Multiplication
```

در مثال زیر درایه واقع در سطر و ستون اول ماتریس M را مقدار ۶ قرار می‌دهیم:

```
1 M[[1, 1]] = 6
2 M[[1, 1]]      Out: 6
```

۷ - ۲ ضرب داخلی و خارجی و محاسبه نورم اول بردار ضرب داخلی (یک عدد است):

```
1 v = {1, 2, 3}
2 v . v           Out: 14
3 Dot[v, v]      Out: 14
```

ضرب خارجی (یک بردار است) :

```
1 Cross[v, v]      Out: {0, 0, 0}
2 v * v            Out: {0, 0, 0}
```

بدست آوردن اندازه نورم اول بردار:

```
1 Norm[v]          Out: Sqrt[14]
```

۷ - ۳ محاسبه دترمینان، جمع درایه‌های قطر اصلی (تربیس ماتریس)، معکوس ماتریس، تولید ماتریس‌های خاص محاسبه دترمینان ماتریس - مثال ۱ :

```
1 M = Table[j^2, {i, 5}, {j, 5}] ;
2 Det[M]        Out: 0
```

محاسبه دترمینان ماتریس - مثال ۲ :

```
1 m = {{1, 2}, {3, 4}}
2 Det[m]        Out: -2
```

محاسبه تربیس ماتریس:

```
1 Tr[m]          Out: 5
```

معکوس ماتریس - مثال ۱ :

```
1 Inverse[m]      Out: {{-2, 1}, {3/2, -(1/2)}}
```

معکوس ماتریس - مثال ۲ :

```
1 Inverse[{{1, 2}, {1, 2}}]      Out: Inverse::sing: Matrix {{1,2},{1,2}} is singular.
```

* در این مثال چون ماتریس singular هست، برنامه ارور می‌دهد. این ماتریس معکوس نیست زیرا که دترمینان آن مساوی با صفر است.

محاسبه رنگ یک ماتریس:

```
1 MatrixRank[{{1, 2}, {1, 2}}]      Out: 1
```

* اگر رنگ یک ماتریس n باشد کامل است و معکوس‌پذیر و در غیر این صورت معکوس‌نپذیر است.
تشکیل ماتریس‌های خاص
ماتریس همانی

```
1 IdentityMatrix[3] // MatrixForm
```

خروجی:

```
Out[•]//MatrixForm=

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

```

ماتریس قطری (همانی نباشد) :

```
1 DiagonalMatrix[{1, 2, 3}] // MatrixForm
```

خروجی:

Out[•]//MatrixForm=

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{pmatrix}$$

۴ - حل دستگاه خطی همگن و ناهمگن

```
1 LinearSolve[{{1, 1}, {1, -1}}, {0, 0}] Out: {0, 0}
2 LinearSolve[{{1, 1}, {1, -1}}, {1, 0}] Out: {1/2, 1/2}
```

مقادیر و بردارهای ویژه یک ماتریس:

```
1 M = {{1, 2}, {3, 4}};
2 Eigenvalues[M] Out: {1/2 (5 + Sqrt[33]), 1/2 (5 - Sqrt[33])}
3 Eigenvectors[M] Out: {{1/6 (-3 + Sqrt[33]), 1}, {1/6 (-3 - Sqrt[33]), 1}}
```