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

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

تنها متغیر a حذف می‌شود.

```
Clear[a]
```

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

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

* همیشه باید در اول هر برنامه از دو کد Remove و Clear 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: $Evaluation \rightarrow EvaluateNotebook$
دستور برای مقداردهی یک متغیر:

```
Variable_name = Value
```

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

```
a = 2
```

مشکلی که این نوع مقداردهی دارد این است که بعد از اجرا سلولی که این کد درون آن است، این مقدار درخروجی 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:

```
a = 2;
```

روش دیگری که موجود است:

```
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:
Evaluation Quit kernel Local
جواب سوال پرسیده شده: Quit
* هر دستور در متمتیکا باید در براکت ([ ]) نوشته شود:

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

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

```
a = 5
b = 8
```

دستور جمع:

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

دستور تفریق:

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

دستور ضرب:

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

دستور تقسیم (روش اول):

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

دستور تقسیم (روش دوم):

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

نحوه خروجی گرفتن با دقت موردنظر:

```
N[expression]                        e.g: N[1/3]        Out: 0.333
N[expression , accuracy]       e.g: N[1/3 , 10]        Out: 0.3333333333
```

برای نوشتن این روش پس از نوشتن صورت تقسیم (در این مثال (Ctrl + / a را می‌زنیم.
روش تایپ کردن فرمول‌های خاص:
$Palettes \rightarrow BasicMathAssistant$
استفاده از حروف یونانی و اسمبل‌ها در این قسمت نیر 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، مثال:

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

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

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

جلسه دوم : معرفی توابع
تابع رادیکال:

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

تابع ساده سازی (Simplify) :

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

در این مثال نماینده آخرین خروجی است.
تابع نمایی:

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

تابع جزء صحیح:

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

تابع قدرمطلق:

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

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

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

تابع فاکتوریل (روش اول) :

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

تابع فاکتوریل (روش دوم) :

```
x!                Out: 120
```

تابع لگاریتم:

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

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

```
Log[E]              Out: 1
```

تغریف پایه الگوریتم:

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

توابع مثلثاتی:

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

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

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

توابع هیپربولیک

```
Sinh[x]
Cosh[x]
Coth[x]
```

تابع تولید اعداد تصادفی
برای اعداد حقیقی:

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

در مثال اول پنج عدد تصادفی در بازه ۲ ، ۲- انتخاب می‌شوند.
در مثال دوم دو دسته پنج تایی عدد تصادفی در بازه ۲ ، ۲- انتخاب می‌شوند.
برای اعداد صحیح:

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

پنج عدد صحیح تصادفی در بازه ۲ ، ۲- انتخاب می‌شوند.
تبدیل عدد به عامل های اول:

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

به توان رساندن:

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

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

کوچک‌ترین مضرب مشترک

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

بزرگترین تقسیم الیه مشترک

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

باقی‌مانده

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

خارج قسمت

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

ترکیب

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

دلتار دیراک

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

دلتا کراینیکر

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

اعداد مختلط:

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

برای تعریف کردن قسمت موهومی و واقعی اعداد مختلط:

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

مزدوج گیری (مختلط) :

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

ریفاین کردن

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

تعریف تابع دلخواه:

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

برای تعریف توابع چند متغیره:

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

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

```
(x + 1)^3              Out: (1 + x)^3
Expand[(x + 1)^3]      Out: 1 + 3 x + 3 x^2 + x^3
(x + 1)^3 // Expand    Out: 1 + 3 x + 3 x^2 + x^3
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
+ 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:

```
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

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

به مثال زیر توجه کنید:

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

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

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

تجزیه کسر به کسرهای جزئی

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

بسط عبارات مثلثاتی و تبدیل آن‌ها به عبارت نهایی

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

ساده سازی عبارت‌های مثلثاتی

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

بسط عبارت‌های مثلثاتی

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

تبدیل عبارت‌های مثلثاتی درجه‌های بالاتر به عبارت خطی

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

تبدیل عبارت‌های مثلثاتی به نمایی و برعکس

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

ساده سازی عبارت‌ها
نکته: در نرم افزار متمتیکا در دو حالت زیر ضرب تعریف می‌شود:
۱) بین دو عبارت علامت ضرب " * " قرار گیرد.
۲) بین دو عبارت اسپیس گذاشته شود.

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

نکته ۱: FullSimplify از Simplify قوی‌تر 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 (مجموعه‌ها)

```
Sum[Sin[i x], {i, 1, 5}]      Out: Sin[x] + Sin[2 x] + Sin[3 x] + Sin[4 x] + Sin[5 x]
Sum[i, {i, 1, n}]             Out: 1/2 n (1 + n)
Sum[i^2, {i, 1, n}]           Out: 1/6 n (1 + n) (1 + 2 n)
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 می‌توان این نماد را رسم کرد.

خروجی کدهای تصویر بالا:

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

حاصل ضرب:

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

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.

$$\prod_{i=1}^{5} i$$

خروجی کد تصویر بالا:

```
Out: 120
```

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

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

خروجی کد بالا به شکل زیر می‌باشد:

Out[•]= $\lim_{x \to x0} f[x]$

Out[•]//TraditionalForm=
$$\lim_{x \to x0} f(x)$$

مثالی دیگر:

```
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.

```
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

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

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

```
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:

```
Derivative[n][f][x + 1 ]
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 موردنظر براش مشتق گرفتن، پارامتر x متغیری که می‌خواهیم طبق آن مشتق گیری انجام دهیم و پارامتر n تعداد مشتق است.

به مثال زیر توجه کنید:

```
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 مشتق‌گیری قرار داده شده است. توجه شود که پارامتر n در این مثال مشخص نشده و به پیش‌فرض ۱ در نظر گرفته می‌شود.

مشتق جزئی:

```
D[f[x, y], x, y]
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:

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

بخش سوم - انتگرال گیری

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

که خروجی آن در ترمینال به صورت زیر نمایش داده می‌شود:

$$Out[\ ]= \int f[x]\, dx$$

یک مثال از انتگرال نامعین:

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

انتگرال دوگانه نامعین:

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

که خروجی آن در ترمینال به صورت زیر نمایش داده می‌شود:

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

یک مثال از انتگرال دوگانه نامعین:

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

انتگرال معین:

```
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:

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

انتگرال‌های عددی:

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

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

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

خروجی به شکل زیر خواهد بود:

Out[•]= $\int_0^4 \mathtt{Sin[Sin[x]]}\, dx$

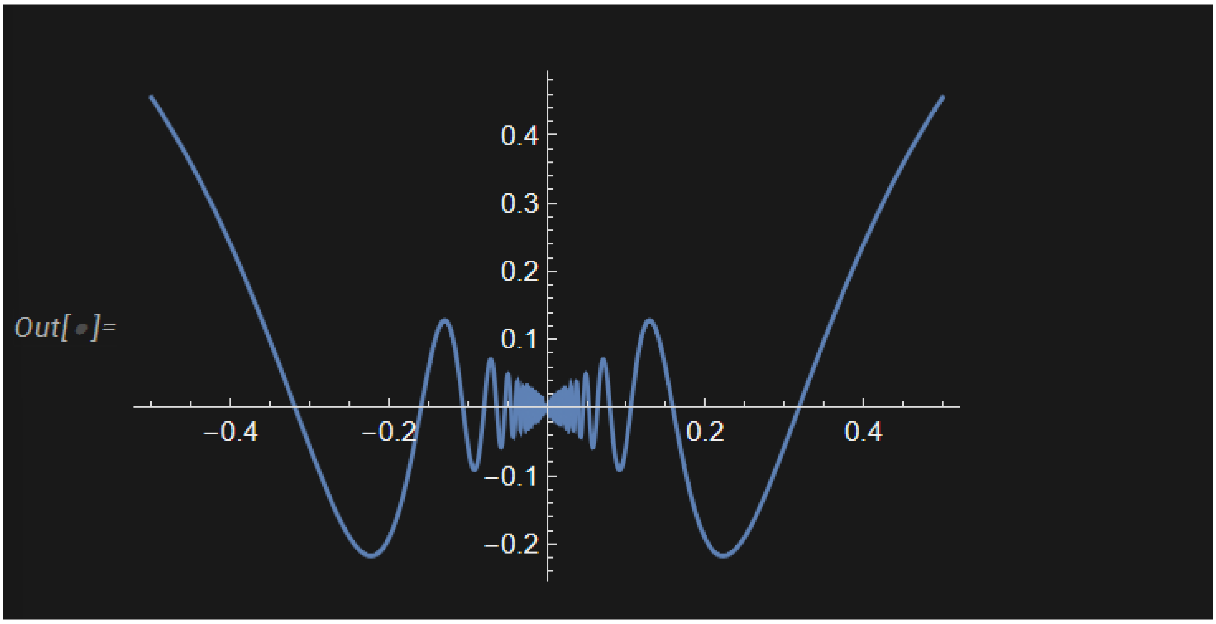
دلیل آن بر این است که این انتگرال به‌صورت بسته با توابع ابتدایی بیان‌شدنی نیست. همین است که Integrate در Mathematica معمولاً جواب نمادین برنمی‌گرداند.
یک مثال دیگر:

```
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:

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

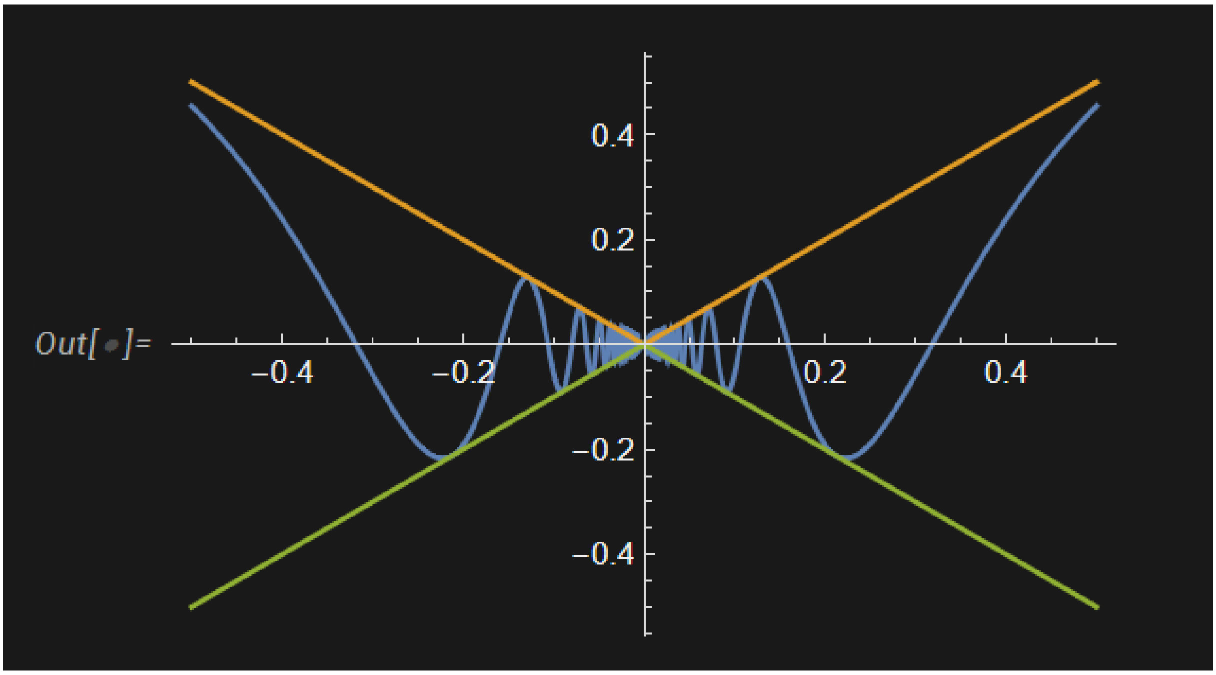
خروجی:



رسم سه تابع در یک دستگاه مختصات:

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

خروجی:



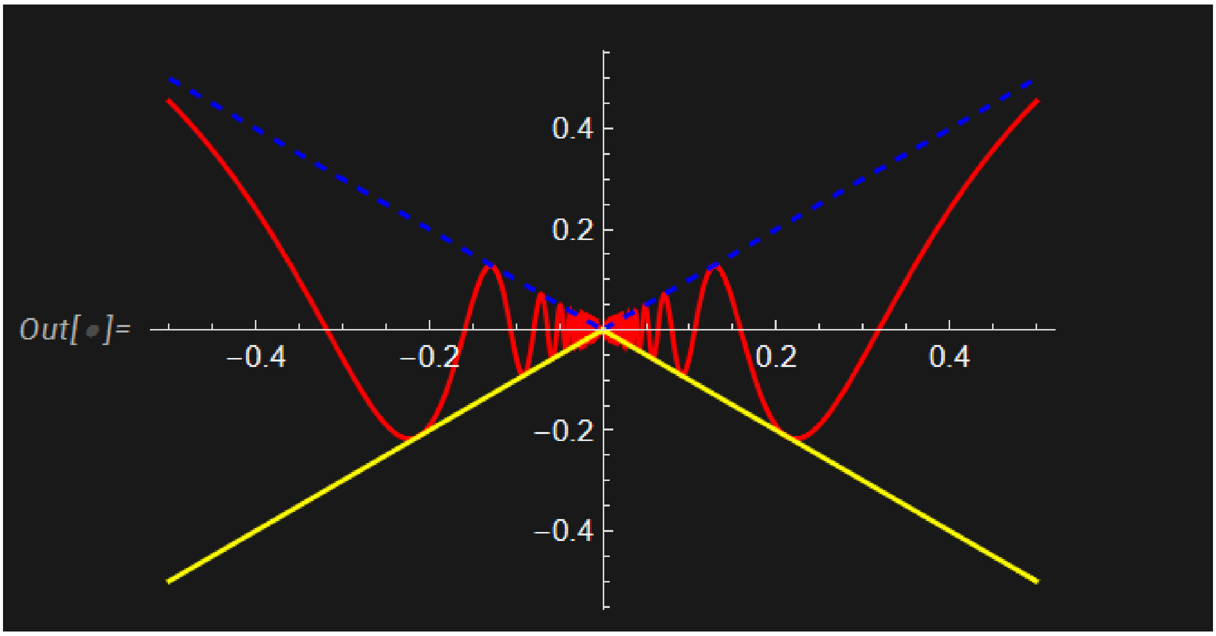
همانطور که مشاهده می‌شود هر سه تابع در یک دستگاه مختصات با رنگ‌های متفاوت رسم می‌شود. برای تغییر رنگ و استایل توابع، در ادامه دستور قبل کد زیر نوشته می‌شود:

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

کد کامل با استایل اضافه شده:

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

خروجی:



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

نکته ۲ : بعد از plotها نمی‌توانیم از ؛ استفاده کنیم، زیرا plot یک تابع است و ذخیره سازی آن ممکن نیست و صرفاً برای نمایش استفاده می‌شود.

برای راهنما قرار دادن برای نمودار، بعد از تغییر رنگ در همان کروشه از دستور زیر استفاده می‌نماییم:

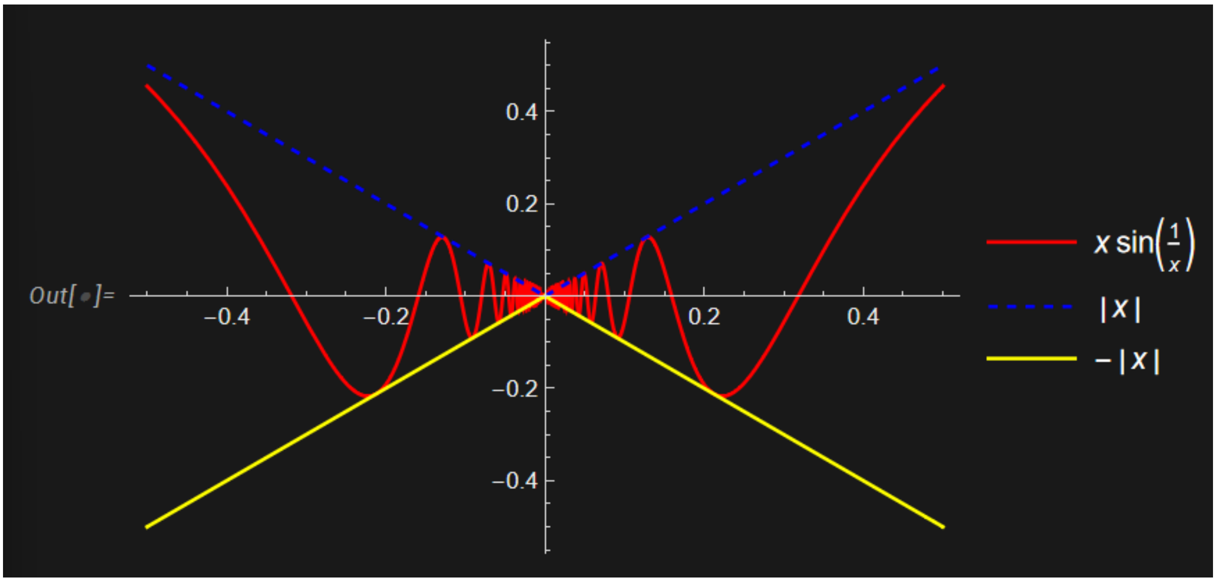
```
PlotLegends -> "Expressions"
```

*توضیحاتی درباره نمودار نمایش داده خواهد شد.

کد کامل با بخش اضافه شده:

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

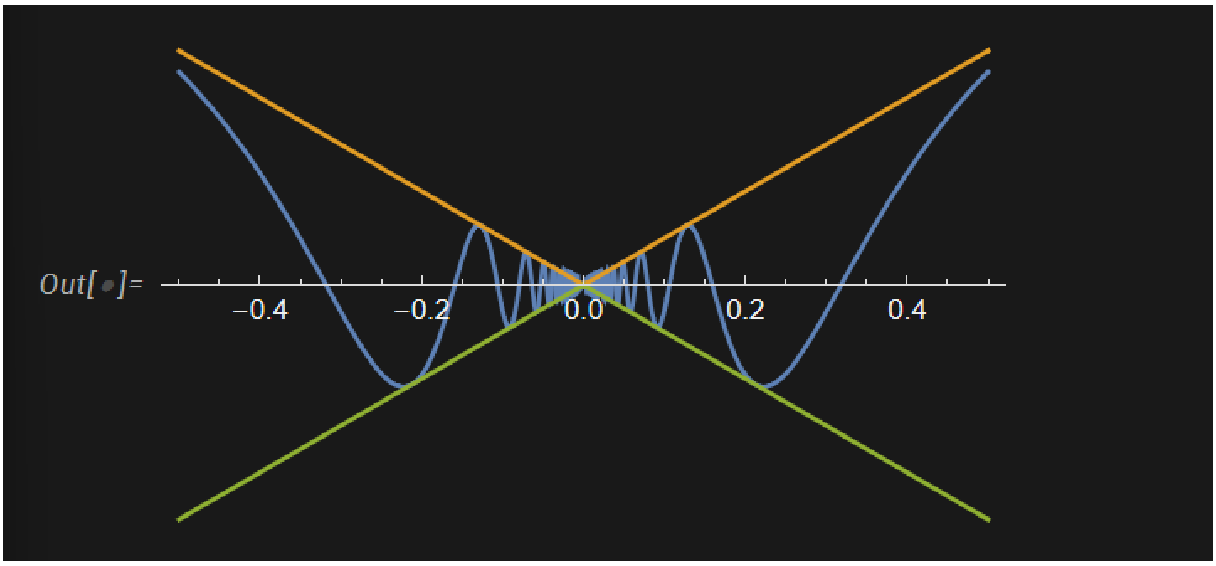
خروجی:



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

```
Plot[{x*Sin[1/x], Abs[x], -Abs[x]}, {x, -1/2, 1/2},
 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 False قرار دهیم، محور x نیز نمایش داده نمی‌شود.

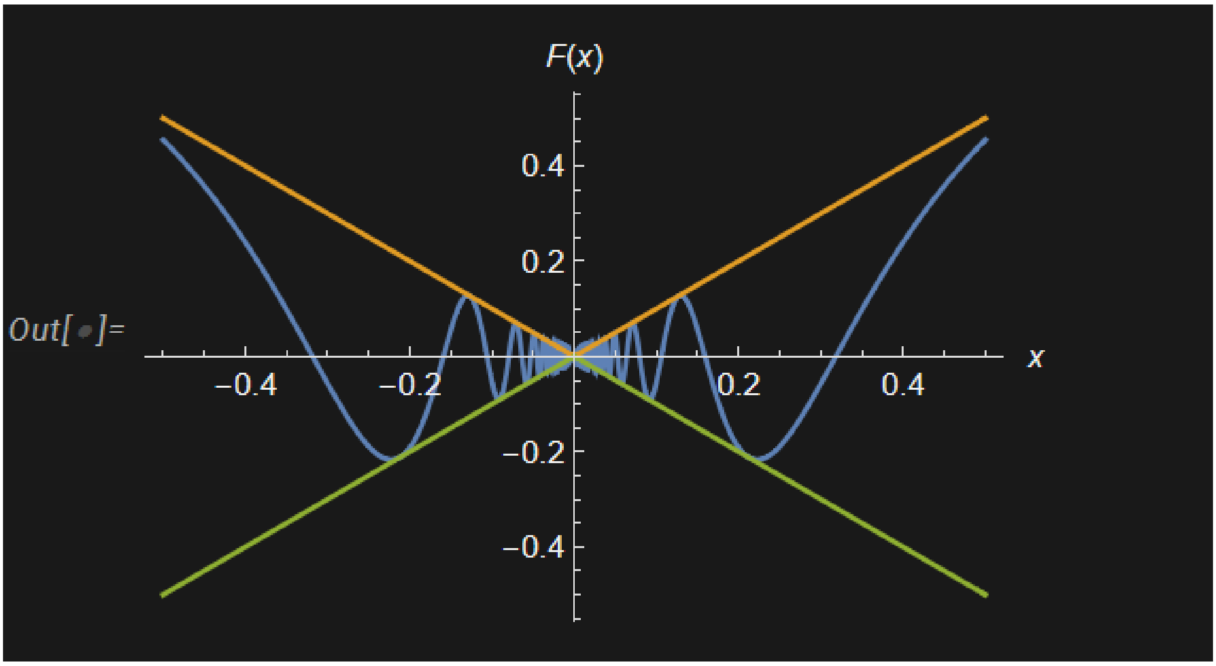
برای نام گذاری برای محورها از دستور زیر استفاده می‌کنیم:

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

کد کامل با بخش اضافه شده:

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

خروجی:



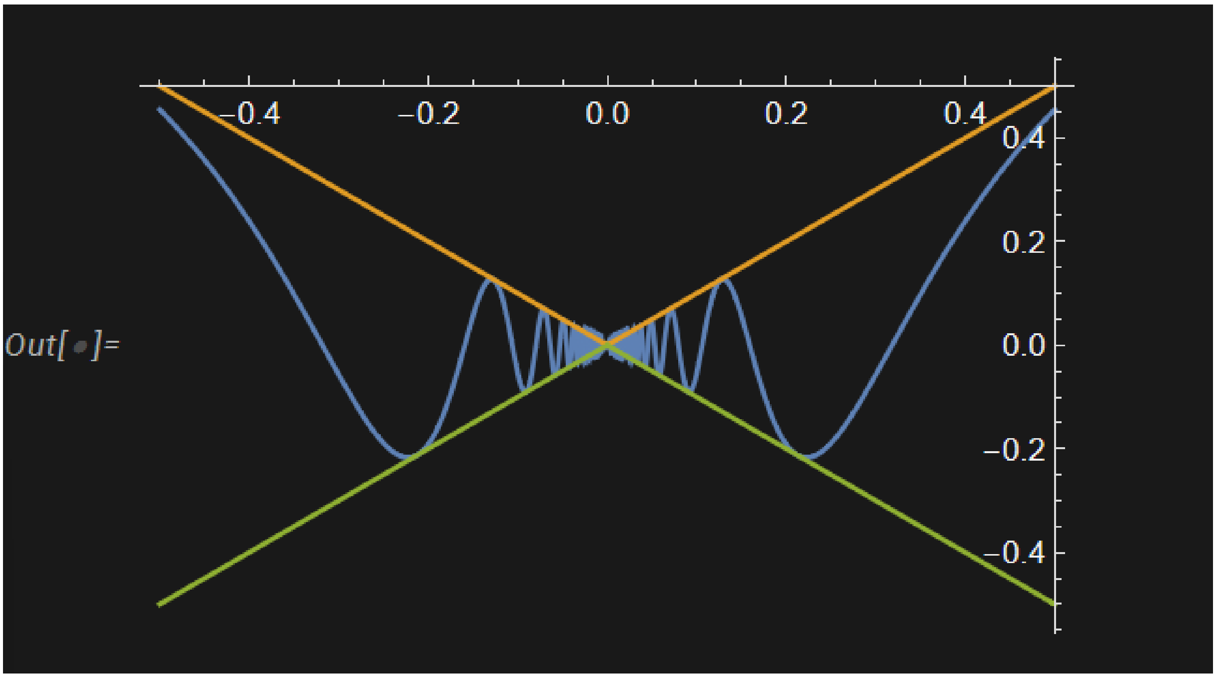
نکته: در این مثال، نام محور y به محور F(x) تغییر داده شده.
برای جابه‌جایی نمودار از دستور زیر استفاده می‌کنیم:

```
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 جابه‌جا شود.
کد کامل با بخش اضافه شده:

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

خروجی:



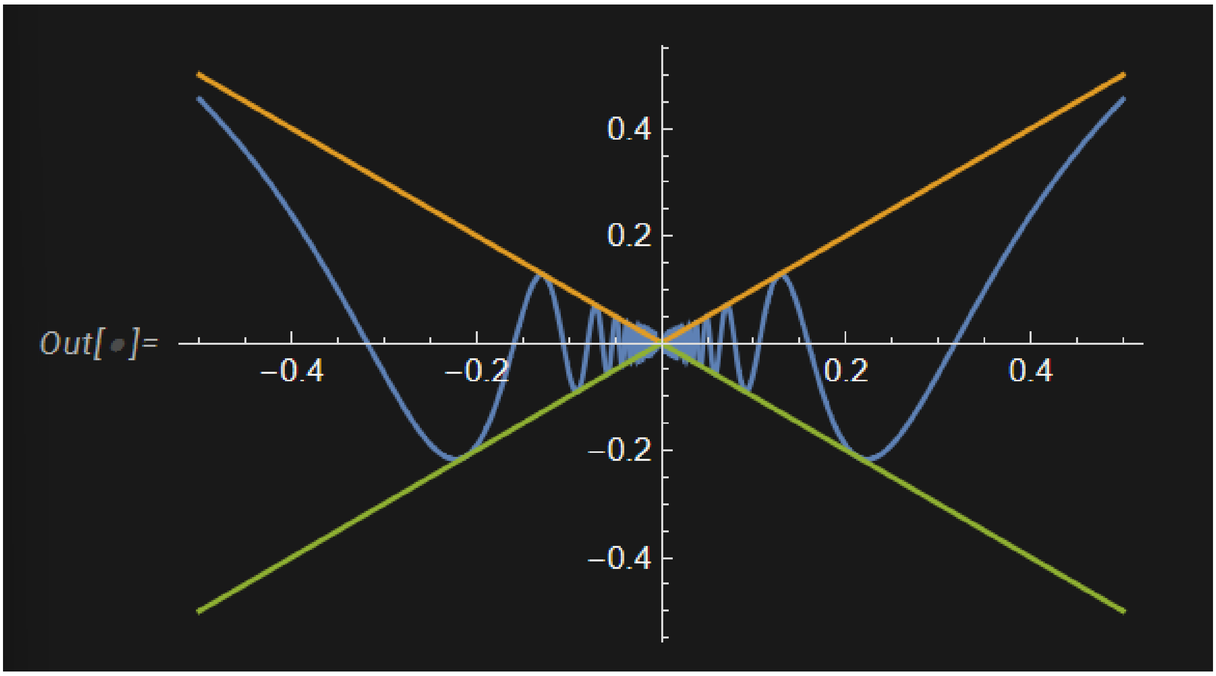
اگر نمودار کامل نمایش داده نشود، می‌توانیم از دستور زیر برای نمایش کامل آن استفاده کنیم:

```
PlotRange -> Full
//or
PlotRange -> All
```

کد کامل با بخش اضافه شده:

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

خروجی:



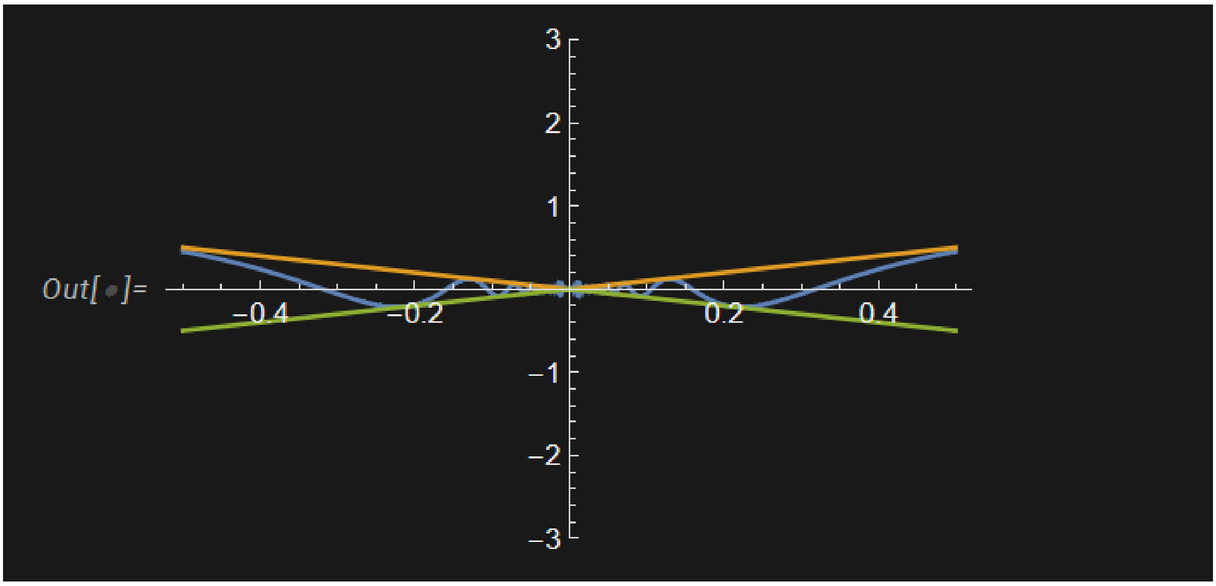
و اگر بخواهیم تا حدی مشخص نمایش داده شود، به جای Full و All آن عدد مشخص را تایپ می‌کنیم:

```
PlotRange -> 3
```

کد کامل با بخش اضافه شده:

```
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:

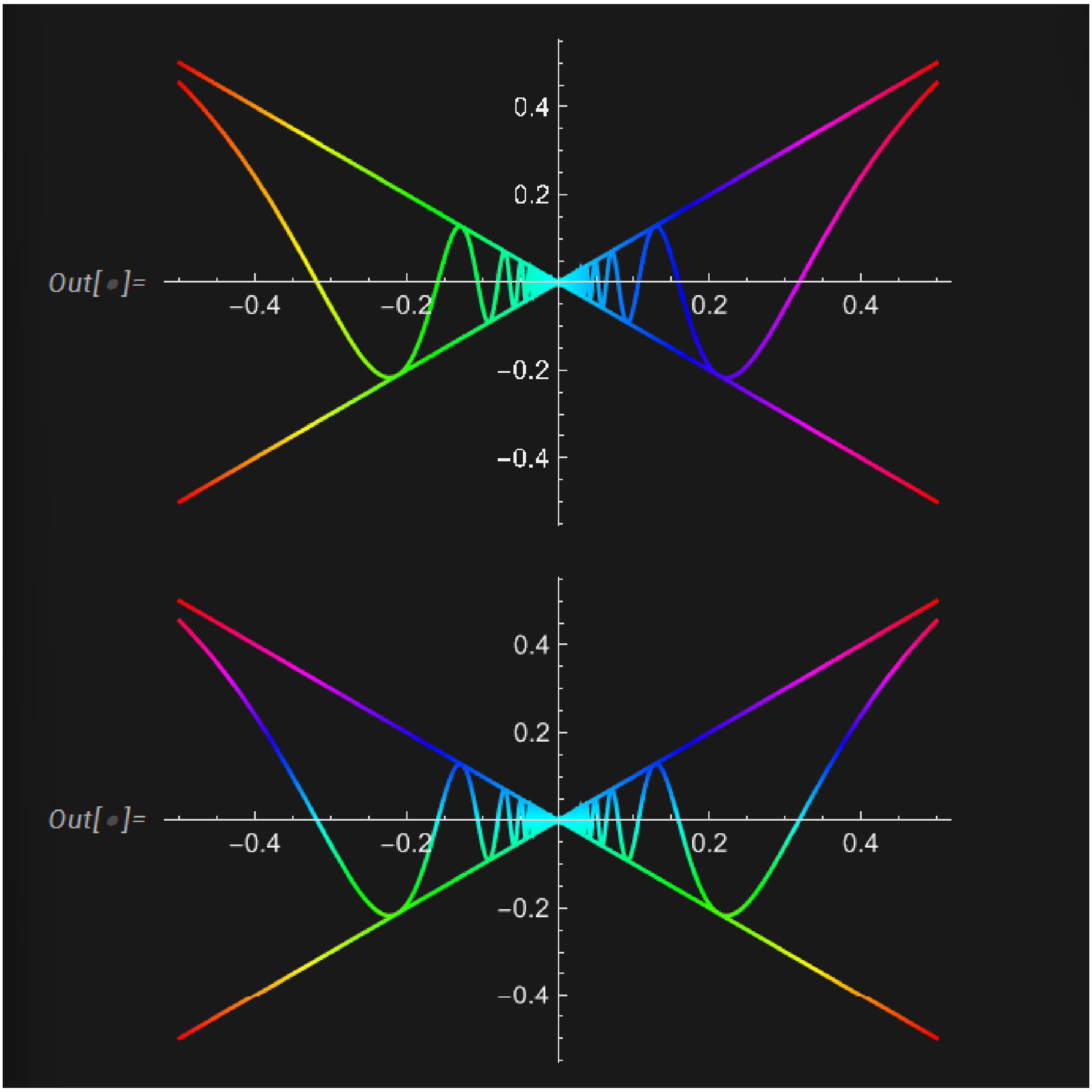
```
ColorFunction -> Function[{x, y}, Hue[y]
```

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:

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

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

خروجی:



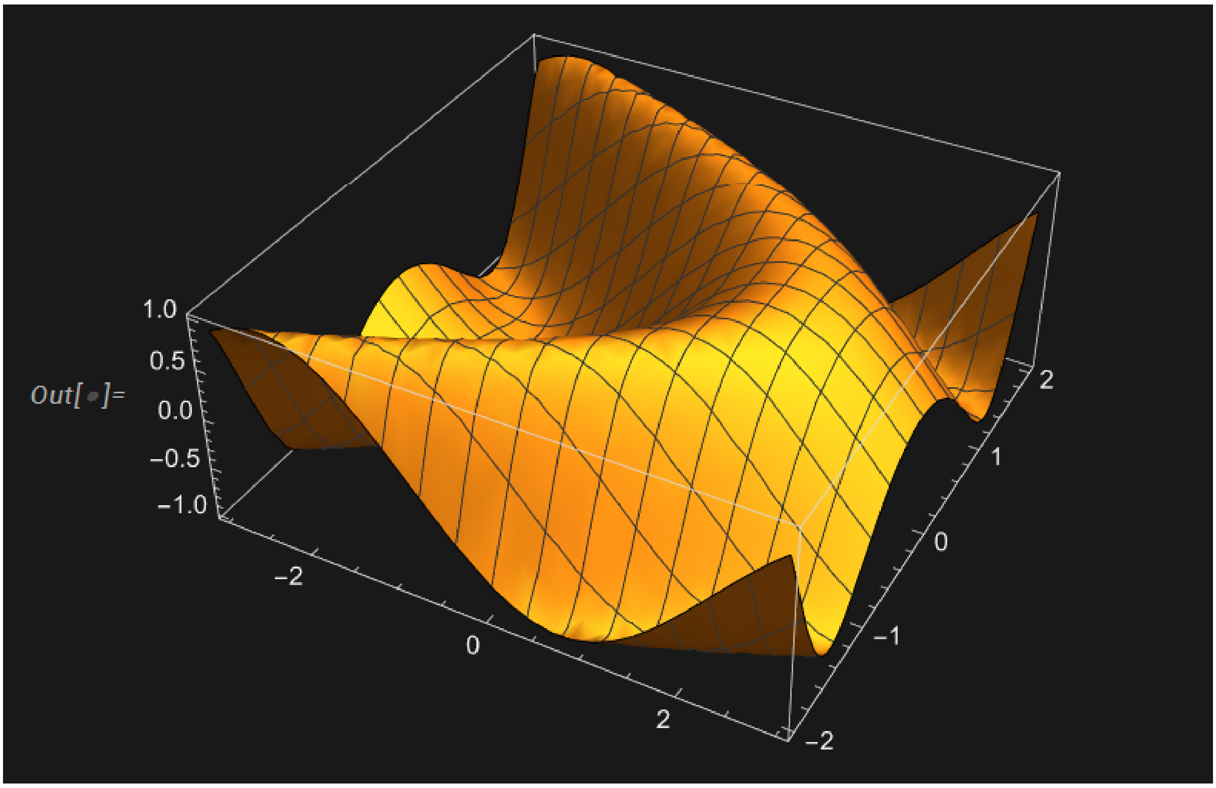
نکته: اگر از ColorFunction استفاده کنیم، نمی‌توانیم در دستور از Legends Plot نیز استفاده کنیم.

۵ - ۲ رسم رویه‌ها

سه بعدی:

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

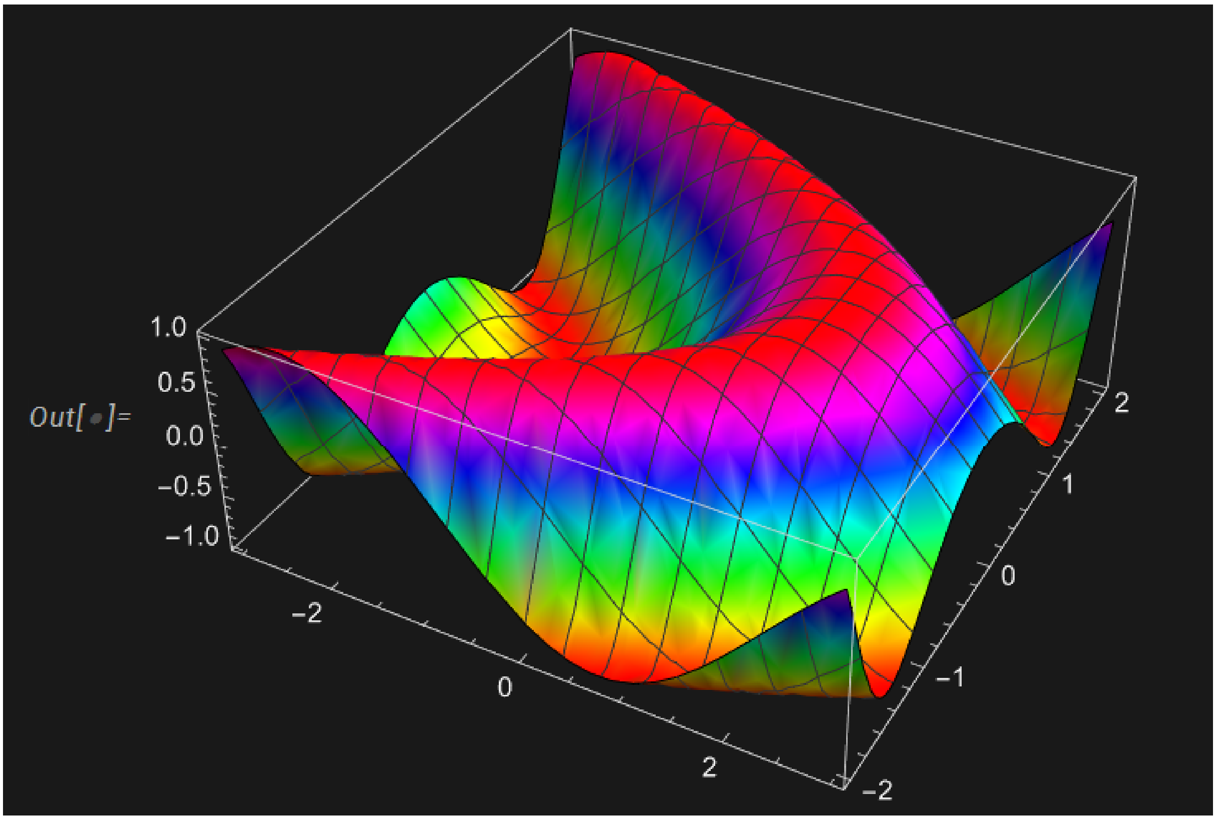
خروجی:



از ColorFinder می‌توانیم در رویه‌های سه بعدی هم استفاده کنیم و بهتر است از 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:



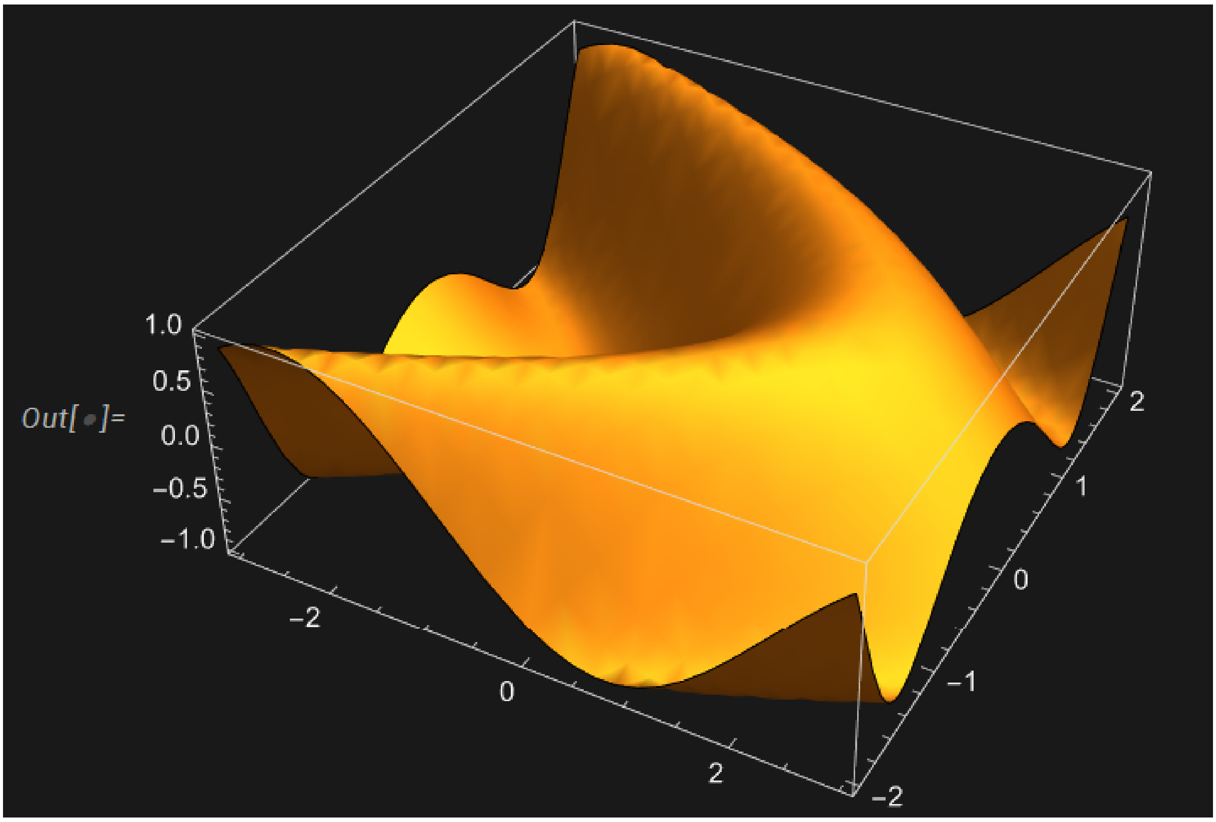
برای حذف چهارخانه‌های روی رویه از دستور زیر استفاده می‌کنیم:

```
Mesh -> None
```

کد کامل با بخش اضافه شده:

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

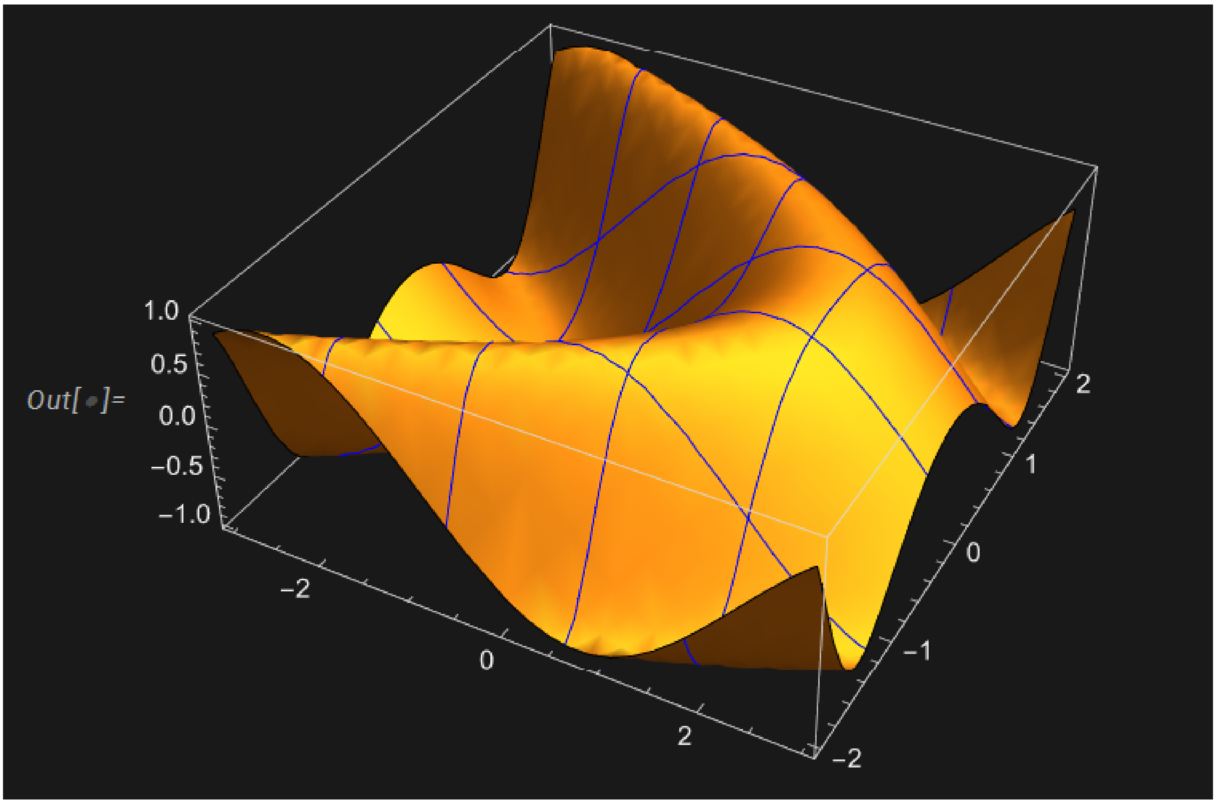
خروجی:



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

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

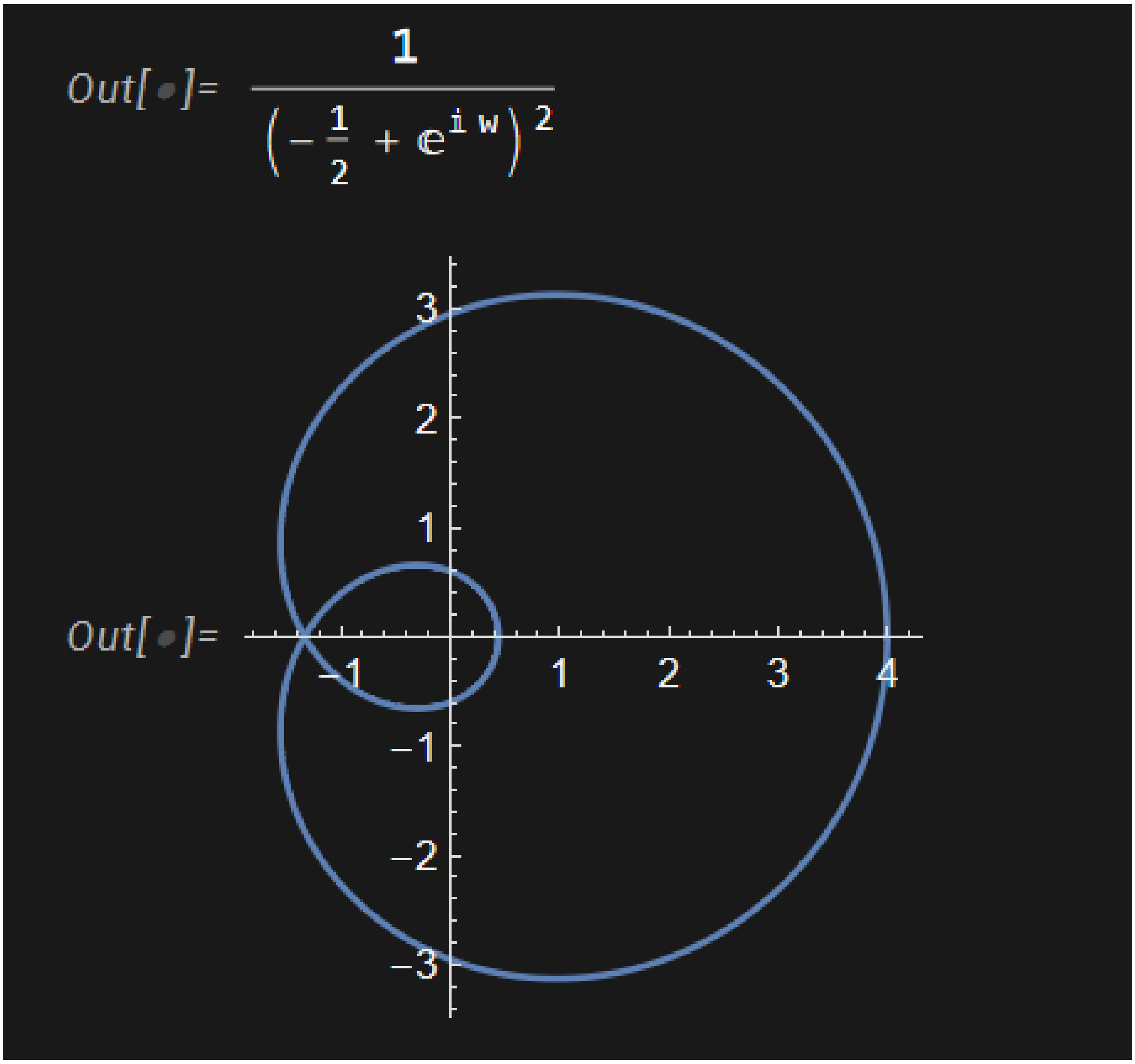
نکته: Blue در این کد رنگ Mesh هستش.
خروجی:



۵ - ۳ رسم توابع پارامتری توابع دو بعدی:

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

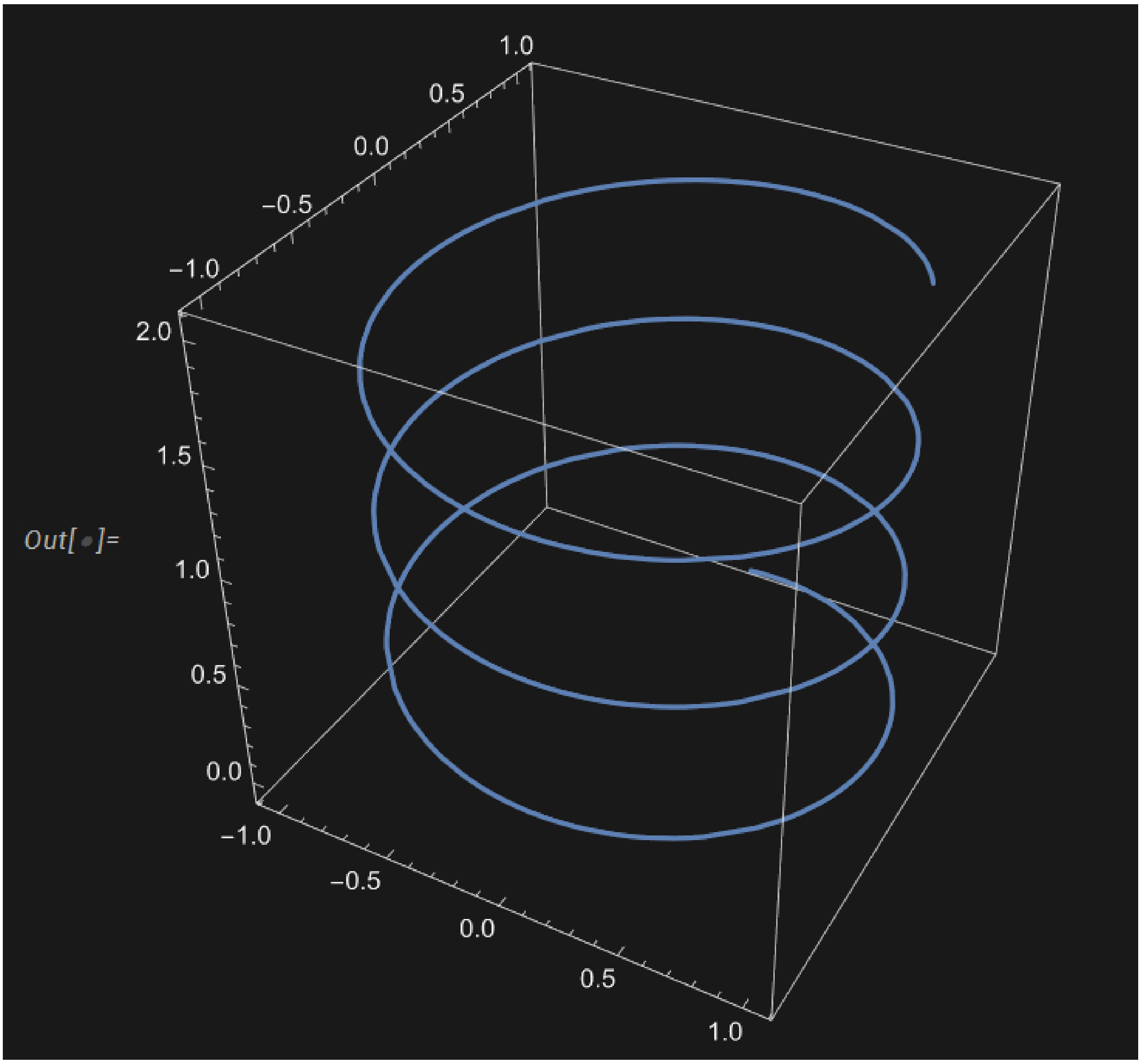
خروجی:



توابع سه بعدی:

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

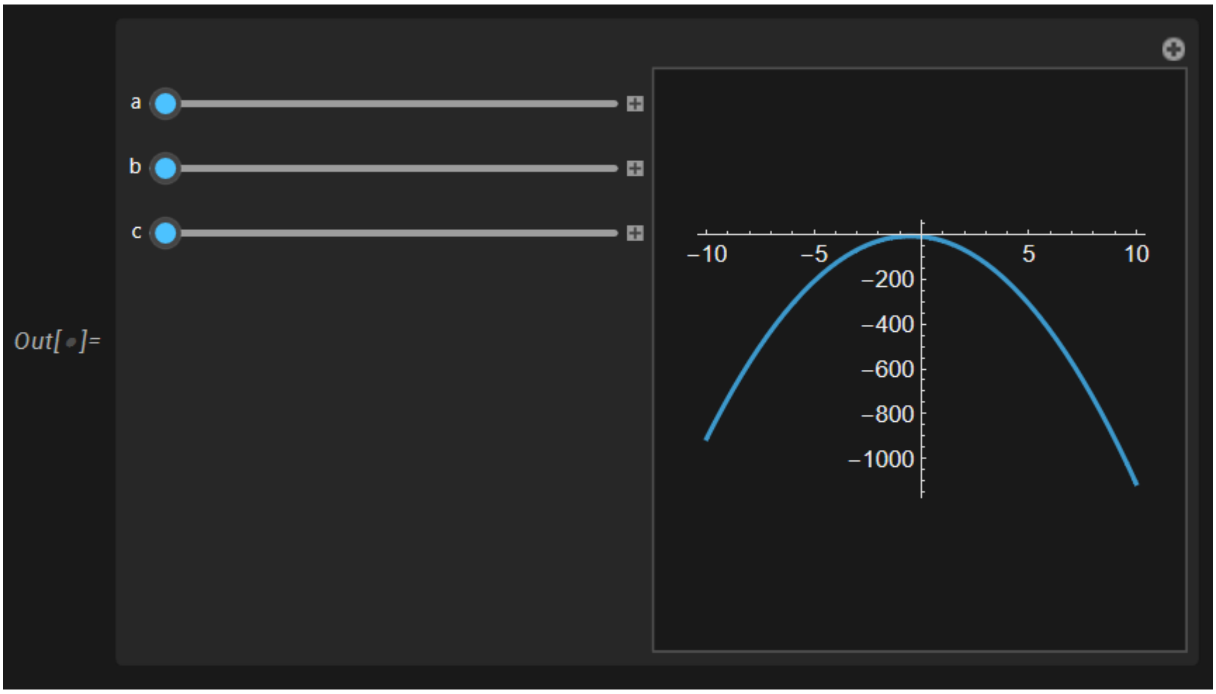
خروجی:



۵ - 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}]
```

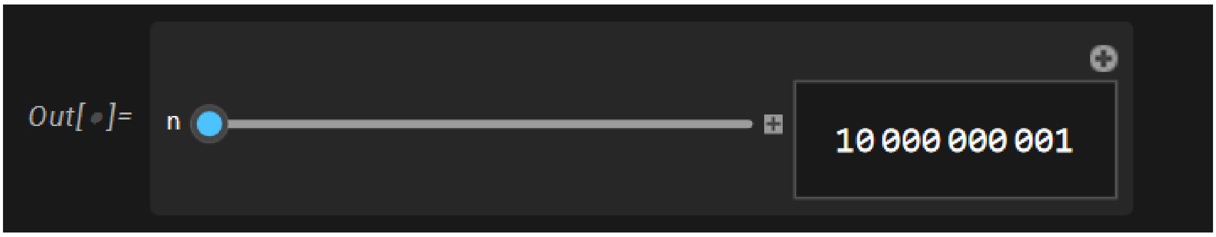
خروجی:



از Manupulate نیز می‌توان برای فاکتورگیری و چندجمله‌ای ها نیز استفاده کرد:

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

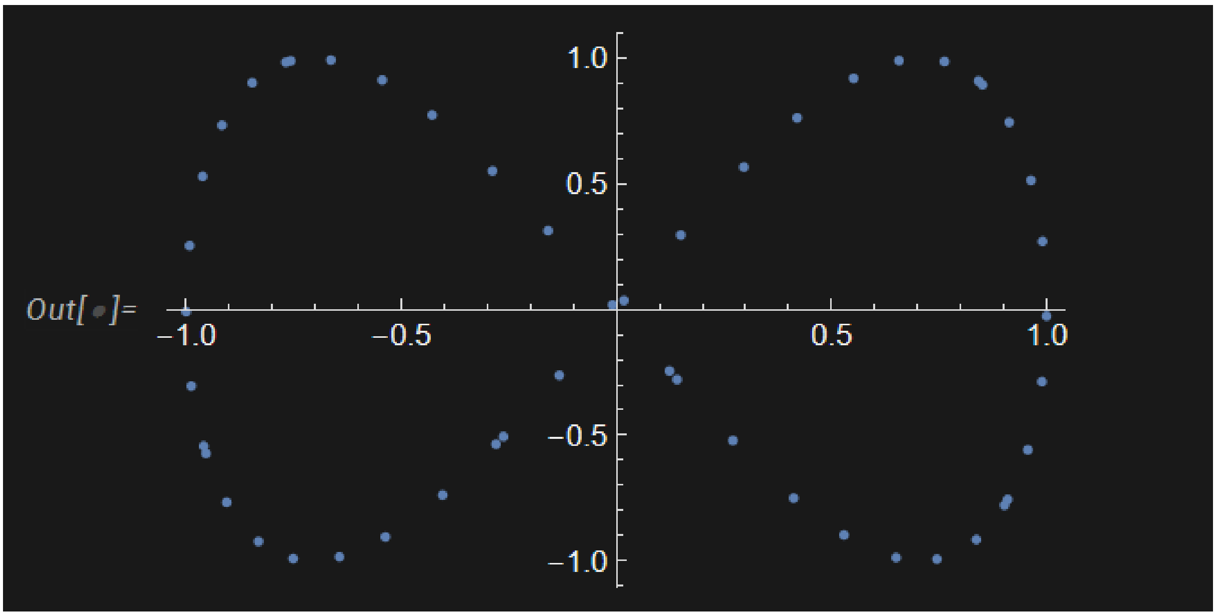
خروجی:



۵ - ۵ رسم توابع با استفاده از نقاط

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

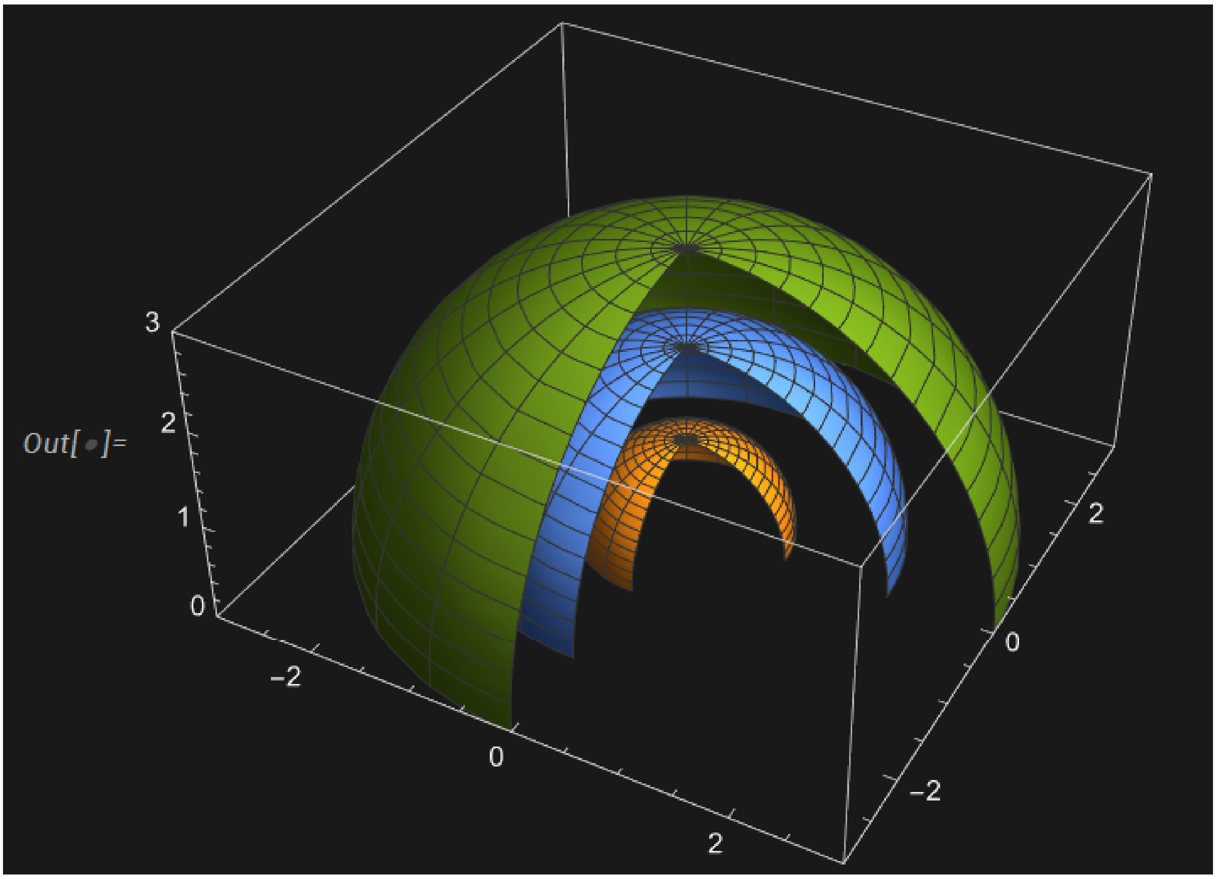
خروجی:



۵ - ۶ رسم منحنی در مختصات‌های مختلف مختصات کروی: مثال اول

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

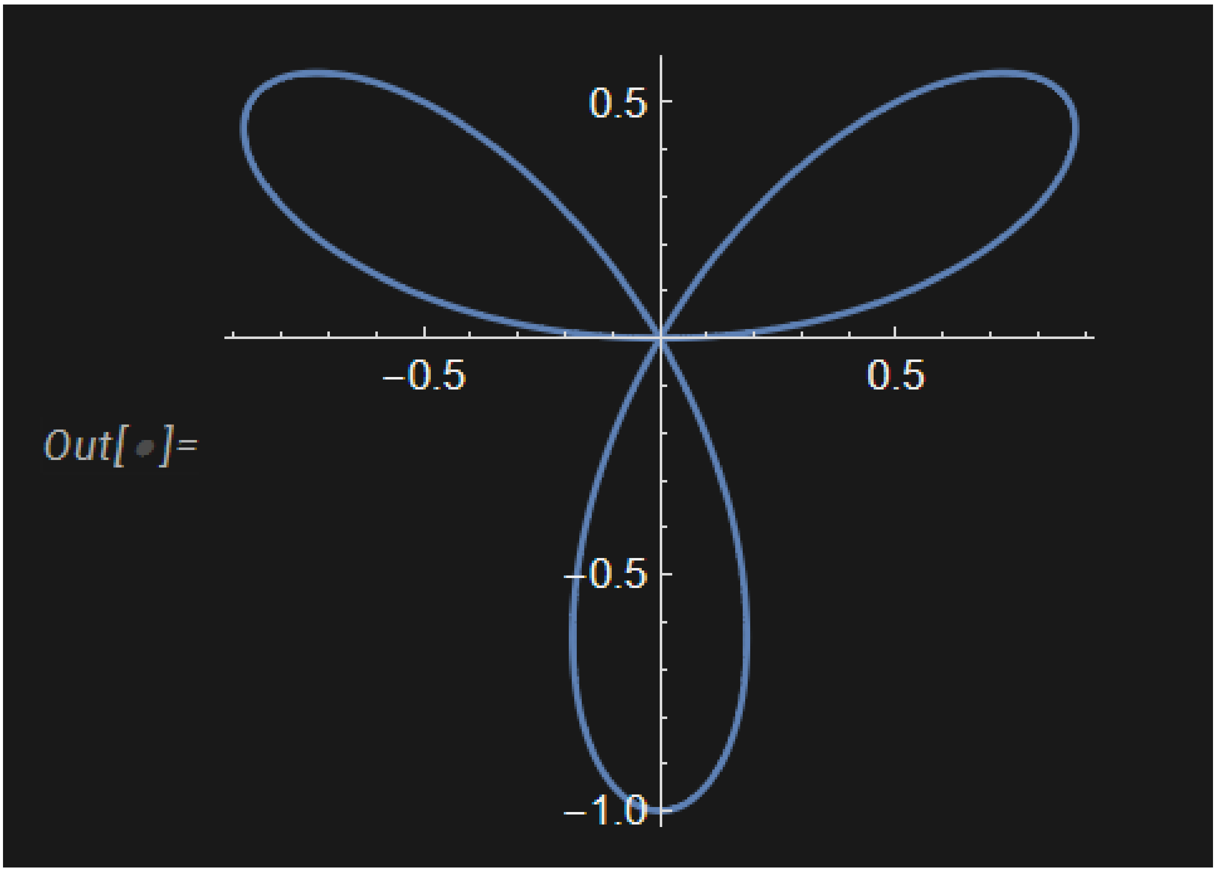
خروجی:



مختصات کروی: مثال دوم

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

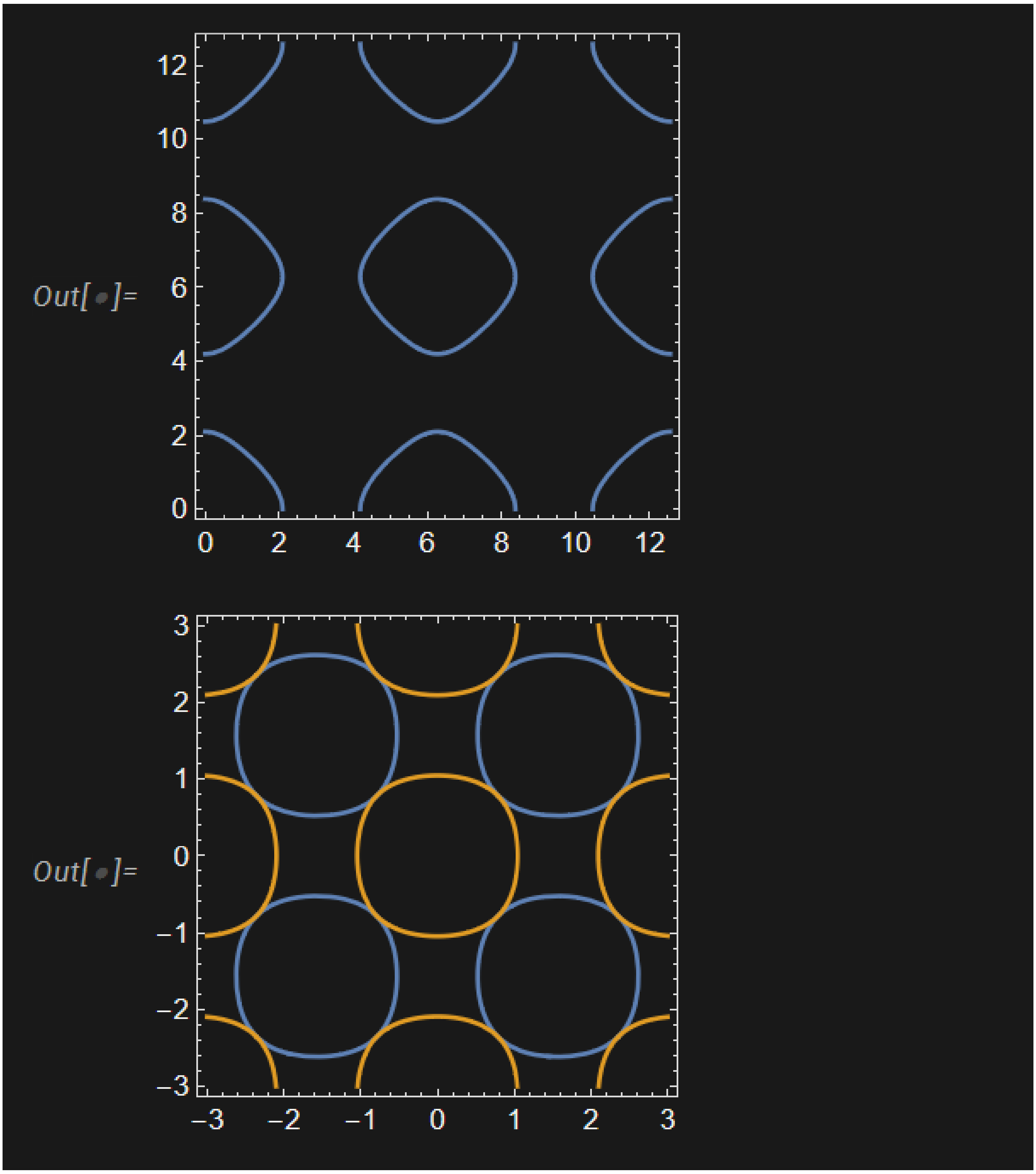
خروجی:



۵ - ۷ رسم محدوده جواب نامعدلات جبری

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

خروجی:



جلسه ششم: حل معادلات ۶ - ۱ حل معادلات جبری

مثال ۱ :

```
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 :

```
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])}
```

* x های جواب معدله را نشان می‌دهد.
مثال ۳ :

```
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 که فقط 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 - ۲ 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 دنبا جواب بگردد از روش زیر استفاده می‌کنیم (روش نیوتون) :

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

* در این مثال، پارامتر دوم که در اینجا ۰ قرار داده شده نقطه شروع ، دو پارامتر بعدی بازه را مشخص می‌کند. در اینجا بازه -۱ تا ۱.
برای بالا بردن دقت محاسبات می‌توانیم از WorkingPrecision استفاده کنیم. به مثال زیر دقت کنید:

```
FindRoot[x^2 - 2, {x, 1}]           Out: {x -> 1.41421}
FindRoot[x^2 - x, {x, 1}, WorkingPrecision -> 60]   Out: {x ->
    1.00000000000000000000000000000000000000000000000000000000000}
```

اگر بخواهیم دقت بسیار بسیار زیاد شود:

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

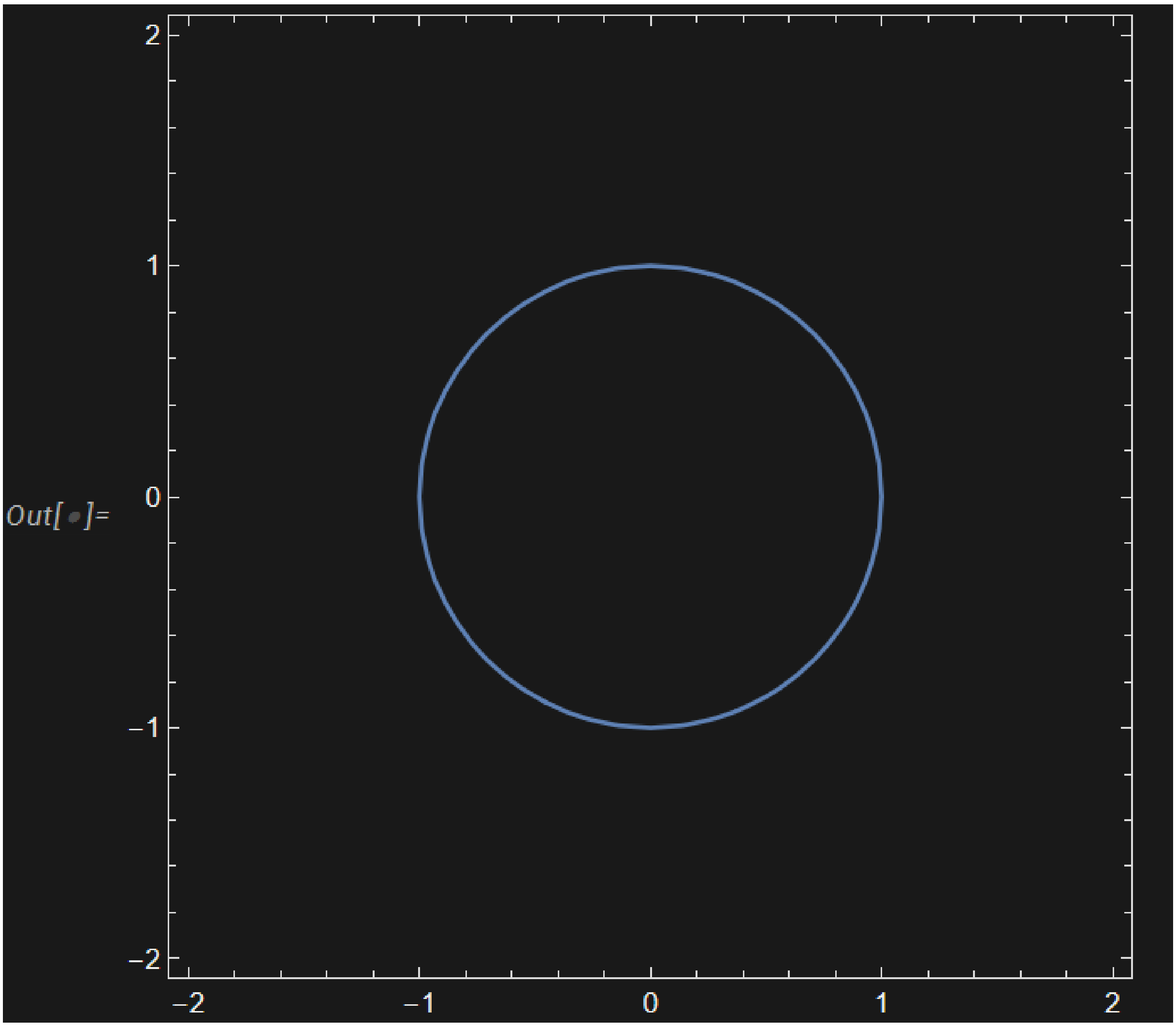
حل نامعادله:

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

رسم دایره:

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

خروجی:



۶ - ۳ معادلات دیفرانسیل معمولی و جزئی

حل معادله دیفرانسیل
- روش تحلیلی (دقیق)
- روش عددی (تقریبی)

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[1] 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])]}}
```

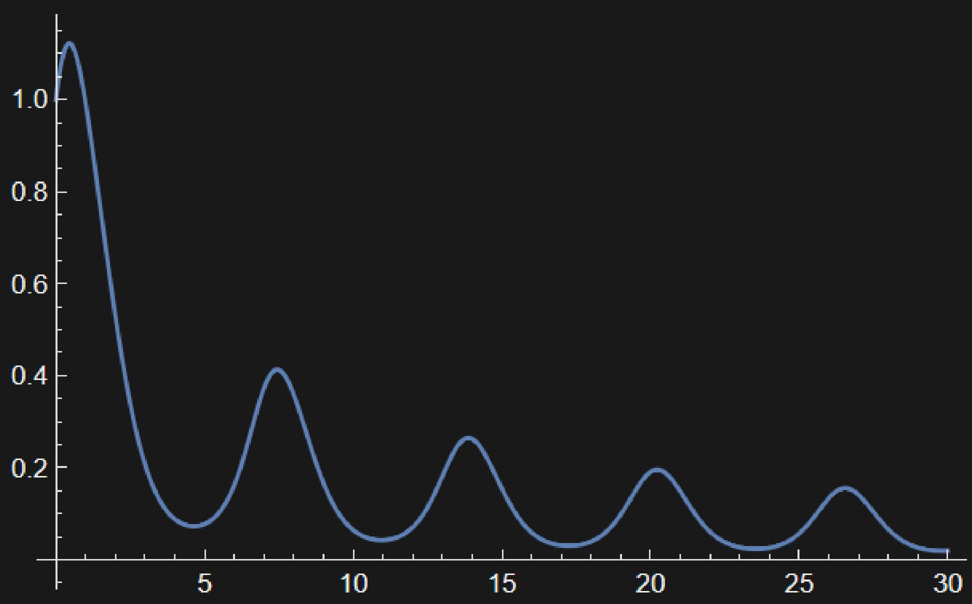
* عبارت y[۰] == ۰ در کد بالا شرط اولیه است.
معادلاتی که می‌توان به روش عددی (تقریبی) حل کرد:

```
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:

Out[ ]= {{y → InterpolatingFunction[Domain: {{0., 30.}} Output: scalar Order: 3 Method: Hermite Periodic: False]}}

Out[ ]=



حل معادلات پارشال:

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

خروجی:

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

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

جلسه هفتم: بردار و ماتریس

نحوه ساخت یک وکتور (بردار) در متمتیکا:

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

نحوه ساخت یک ماتریس:

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

اگر بخواهیم به شکل ماتریس نمایش داده شود:

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

خروجی:

Out[ ]//MatrixForm=

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

روش ساخت یک بردار و ماتریش با دستور Table :

```
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:

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

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

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

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

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

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

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

ضرب خارجی (یک بردار است) :

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

بدست آوردن اندازه نورم اول بردار:

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

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

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

محاسبه دترمینان ماتریس - مثال ۲ :

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

محاسبه تریس ماتریس:

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

معکوس ماتریس - مثال ۱ :

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

معکوس ماتریس - مثال ۲ :

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

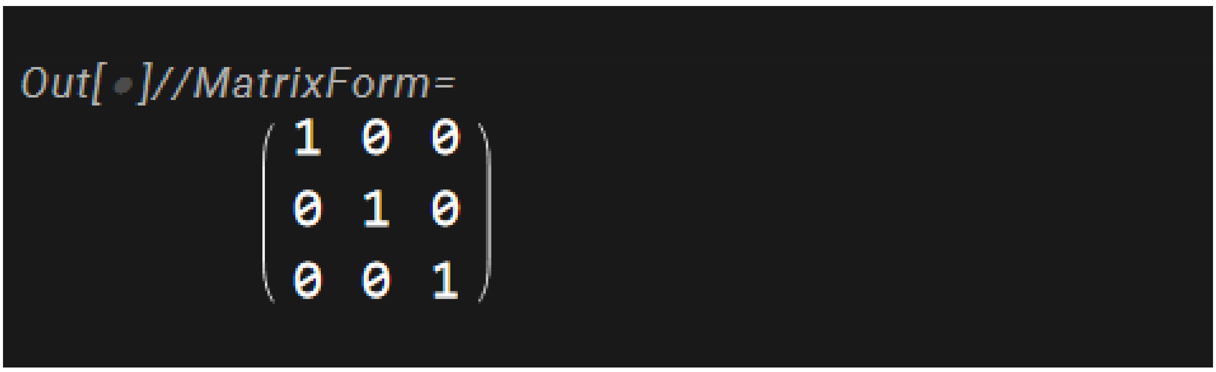
* در این مثال چون ماتریس singular هست، برنامه ارور می‌دهد. این ماتریس معکوس نیست زیرا که دترمینان آن مساوی با صفر است.
محاسبه رنک یک ماتریس:

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

* اگر رنک یک ماتریکس n باشد کامل است و معکوس‌پذیر و در غیر این صورت معکوس‌ناپذیر است.
تشکیل ماتریس‌های خاص
ماتریس همانی

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

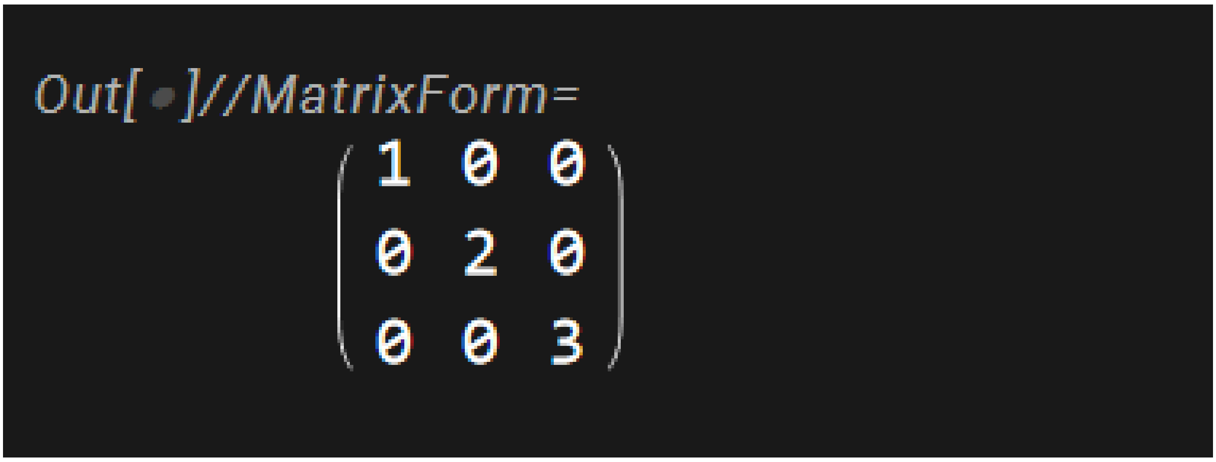
خروجی:



ماتریس قطری (همانی نباشد) :

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

خروجی:



۷ - ۴ حل دستگاه خطی همگن و ناهمگن

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

مقادیر و بردارهای ویژه یک ماتریس:

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