Computer Applications for Engineers

ET 601

Asst. Prof. Dr. Prapun Suksompong

prapun@siit.tu.ac.th

Random Variables

Office Hours: (BKD 3601-7)

Wednesday	9:30-11:30
Wednesday	16:00-17:00
Thursday	14:40-16:00

Example: Roll a dice

- Let X denotes the result.
- This X is called a **random variable (RV)**.
- We can simulate this in MATLAB by `X = randi(6)`.
- There are 6 possible values of X: 1,2,3,4,5,6
  - The set of these number is call a **support** of X.
- Technically, a set S is called a **support** of a random variable X if the probability that $X \in S$ is one.
  - For this example, a bigger set such as $\{1,2,3, \dots, 10\}$ is also a support for X.
  - We usually mean the ***minimal*** support when we say support.
  - When we want to emphasize that the set S is a support of a particular random variable X, we write S_X instead of S.

Discrete Random Variable

- X is a **discrete** random variable if it has a countable support.
  - Recall that countable sets include finites set and countably infinite sets.
- For X whose support is uncountable, there are two types:
  - **Continuous** random variable
  - **Mixed** random variable

Probabilities involving discrete RV

- Back to example of rolling a dice
- The "important" probabilities are

$$P[X=1]=P[X=2]=\cdots=P[X=6]=\frac{1}{6}$$

- In tabular form:

Dummy variable

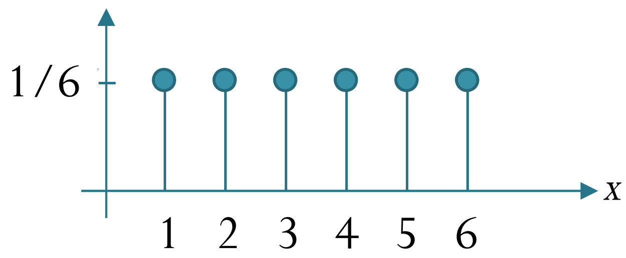
x	$P[X=x]$
1	1/6
2	1/6
3	1/6
4	1/6
5	1/6
6	1/6

- **Probability mass function (PMF)**:

$$p_X(x)=\begin{cases}1/6, & x=1,2,3,4,5,6,\\ 0, & \text{otherwise.}\end{cases}$$

  - In general, $p_X(x)\equiv P[X=x]$

- Stem plot:



Probabilities involving discrete RV

To find P[some condition(s) on X] from the pmf $p_X(x)$ of X:

1. Find the support of X.
2. Look only at values x inside the support.
   Find all x that satisfies the condition(s).
3. Evaluate the pmf at x found in the previous step.
4. Add the pmf values from the previous step.

Back to the dice roll example. Suppose we want to find $P[X > 4]$.

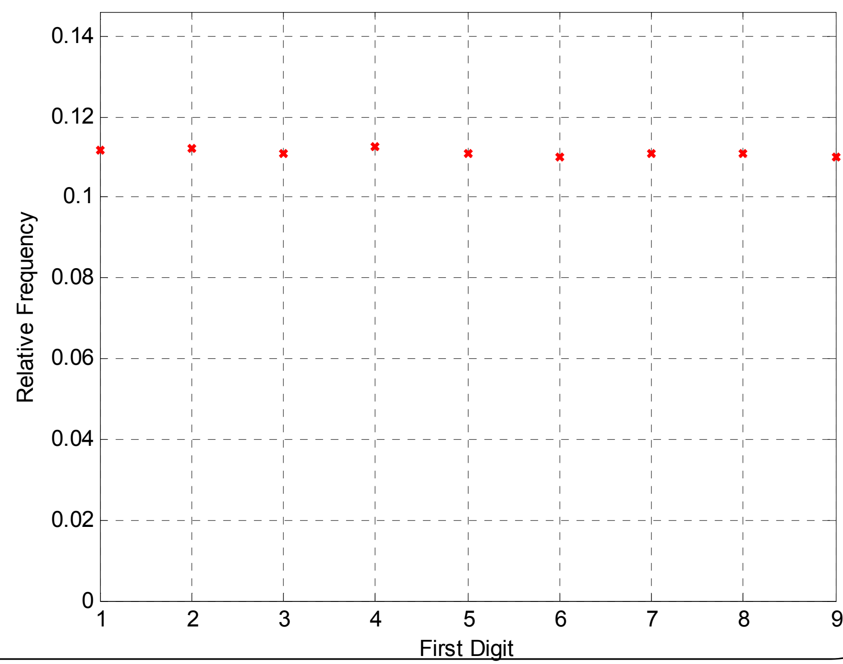
1. The support of X is $\{1,2,3,4,5,6\}$.
2. The members which satisfies the condition ">4" is 5 and 6.
3. The pmf values at 5 and 6 are all 1/6.
4. Adding the pmf values gives $2/6 = 1/3$.

Benford's law: Introduction

- Consider the distribution of the **first (leading) digit** in real-life sources of data.
- Suppose you start reading through a particular issue of a publication like the New York Times or The Economist, and each time you encounter any number (the amount of donations to a particular political candidate, the age of an actor, the number of members of a union, and so on), you record the first digit of that number. Possible first digits are 1, 2, 3, … , or 9. In the long run, how frequently do you think each of these nine possible first digits will be encountered?
- It might be quite natural to assume that all digits are equally likely to show up in most random data sets.

```
X = randi(1e6,1e5,1);
```

560447
845196
901480
639879
449454
41875
365825
41551
976613
706264
164932
88515
452648
820554



Benford's law: Introduction

- One of the following columns contains the value of the closing stock index as of Aug. 8, 2012 for each of a number of countries, and the other column contains fake data obtained with a random number generator.
- Just by looking at the numbers, without considering context, can you tell which column is fake and which is real?

China	2264	3058
Japan	8881	9546
Britain	5846	7140
Canada	11,781	6519
Euro area	797	511
Austria	2053	4995
France	3438	2097
Germany	6966	4628
Italy	14,665	8461
Spain	722	598
Norway	480	1133
Russia	1445	4100
Sweden	1080	2594
Turkey	64,699	35,027
Hong Kong	20,066	42,182
India	17,601	3388
Pakistan	14,744	10,076
Singapore	3052	5227
Thailand	1214	7460
Argentina	2459	2159
⋮	⋮	⋮

http://www.census.gov/population/international/data/countryrank/rank.php

Countries and Areas Ranked by Population:2013

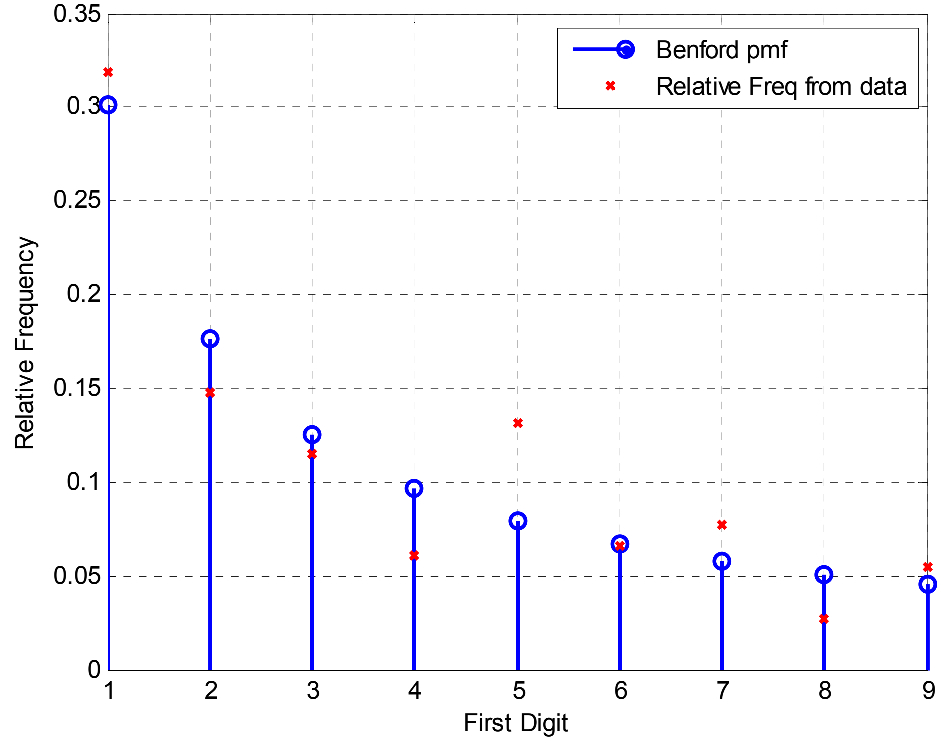
Rank	Country Or Area	Population
1	China	1,349,585,838
2	India	1,220,800,359
3	United States	316,438,601
4	Indonesia	251,160,124
5	Brazil	201,009,622
6	Pakistan	193,238,868
7	Nigeria	172,831,537
8	Bangladesh	163,654,860
9	Russia	142,500,482
10	Japan	127,253,075
11	Mexico	118,818,228
12	Philippines	105,720,644
13	Ethiopia	93,877,025
14	Vietnam	92,477,857
15	Egypt	85,294,388
16	Germany	81,147,265
17	Turkey	80,694,485
18	Iran	79,853,900

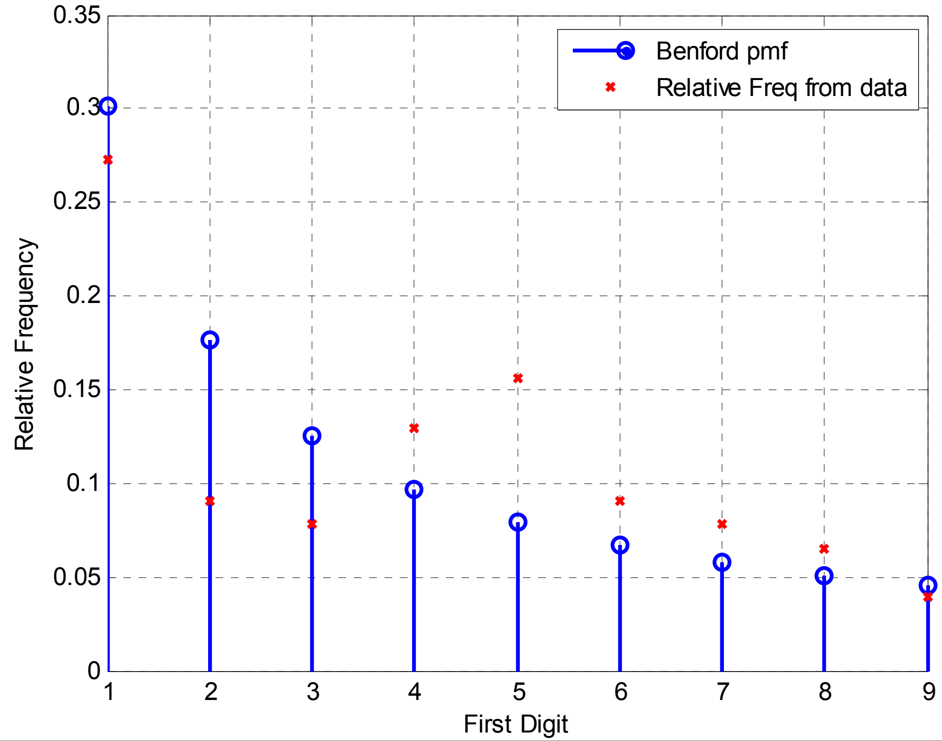
ประกาศสำนักทะเบียนกลาง กรมการปกครอง
เรื่อง จำนวนราษฎรทั่วราชอาณาจักร แยกเป็นกรุงเทพมหานครและจังหวัดต่าง ๆ
ตามหลักฐานการทะเบียนราษฎร ณ วันที่ ๓๑ ธันวาคม ๒๕๕๕

อาศัยอำนาจตามความในมาตรา ๔๕ แห่งพระราชบัญญัติการทะเบียนราษฎร พ.ศ. ๒๕๓๔ จึงประกาศจำนวนราษฎรทั่วราชอาณาจักร แยกเป็นกรุงเทพมหานครและจังหวัดต่าง ๆ ตามหลักฐานการทะเบียนราษฎร ณ วันที่ ๓๑ ธันวาคม ๒๕๕๕ ดังต่อไปนี้

ลำดับ	จังหวัด	จำนวนราษฎร		
		ชาย	หญิง	รวม
	ทั่วประเทศ	๓๑,๗๐๐,๗๒๗	๓๒,๗๕๕,๙๖๘	๖๔,๔๕๖,๖๙๕
๑	กรุงเทพมหานคร	๒,๖๙๐,๗๕๔	๒,๙๘๒,๘๐๖	๕,๖๗๓,๕๖๐
๒	จังหวัดกระบี่	๒๒๑,๙๐๖	๒๒๓,๐๖๑	๔๔๔,๙๖๗
๓	จังหวัดกาญจนบุรี	๔๒๐,๔๖๗	๔๑๗,๘๐๒	๘๓๘,๒๖๙
๔	จังหวัดกาฬสินธุ์	๔๙๐,๖๑๗	๔๙๔,๔๖๗	๙๘๕,๐๘๔
๕	จังหวัดกำแพงเพชร	๓๖๑,๕๑๑	๓๖๖,๐๔๔	๗๒๗,๕๕๕
๖	จังหวัดขอนแก่น	๘๗๙,๓๖๐	๘๙๕,๔๕๖	๑,๗๗๔,๘๑๖
๗	จังหวัดจันทบุรี	๒๕๖,๖๙๐	๒๖๕,๑๒๒	๕๒๑,๘๑๒
๘	จังหวัดฉะเชิงเทรา	๓๓๕,๙๘๓	๓๔๙,๗๓๘	๖๘๕,๗๒๑
๙	จังหวัดชลบุรี	๖๖๘,๗๔๔	๖๙๕,๒๕๘	๑,๓๖๔,๐๐๒
๑๐	จังหวัดชัยนาท	๑๖๐,๗๖๙	๑๗๒,๔๐๓	๓๓๓,๑๗๒
๑๑	จังหวัดชัยภูมิ	๕๖๓,๓๒๓	๕๖๙,๗๑๑	๑,๑๓๓,๐๓๔
๑๒	จังหวัดชุมพร	๒๔๕,๙๗๓	๒๔๙,๓๓๗	๔๙๕,๓๑๐
๑๓	จังหวัดเชียงราย	๕๙๐,๔๔๖	๖๐๙,๙๗๗	๑,๒๐๐,๔๒๓
๑๔	จังหวัดเชียงใหม่	๘๐๖,๗๒๐	๘๔๘,๙๒๒	๑,๖๕๕,๖๔๒
๑๕	จังหวัดตรัง	๓๐๙,๗๒๙	๓๒๒,๑๙๑	๖๓๑,๙๒๐

http://stat.bora.dopa.go.th/stat/y_stat55.html



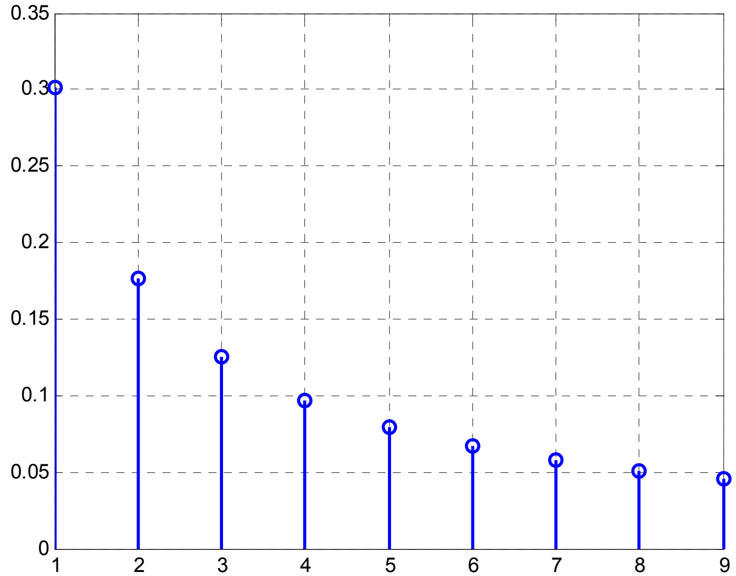


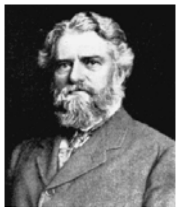
Benford's law

Zero is inadmissible as a first digit.
The signs of negative numbers are ignored.

- The distribution of the **first digit** in many (but not all) real-life sources of data.

$$p_X(x)=\begin{cases}\log_{10}\left(1+\dfrac{1}{x}\right), & x=1,2,3,\ldots 9,\\ 0, & \text{otherwise.}\end{cases}$$





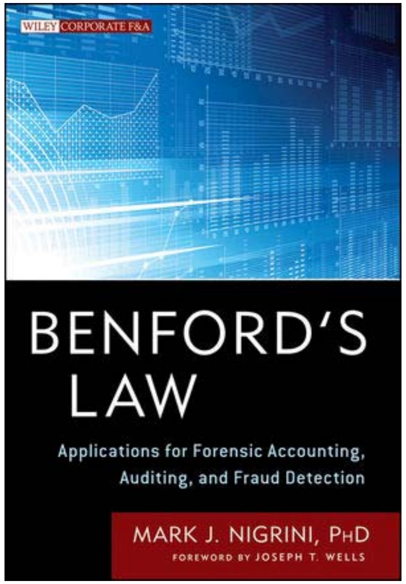
- Named after an American physicist Frank Benford, who stated it in 1938, although it had been previously stated by Simon Newcomb in 1881.

[Benford, "The law of anomalous numbers", Proceedings of the American Philosophical Society, vol. 78, pp. 551–572, 1938.]

- There is a large bias towards the lower digits, so much so that nearly one-half of all numbers are expected to start with the digits 1 or 2.

Benford's law

- Applicable to a wide variety of data sets, including electricity bills, street addresses, stock prices, population sizes, death rates, lengths of rivers, physical and mathematical constants.
- It tends to be most accurate when values are distributed across multiple orders of magnitude.
- Today, Benford's law is routinely applied in several areas in which naturally occurring data arise.
- Perhaps the most practical application of Benford's law is in **detecting fraudulent** data (or unintentional errors) in accounting reports, and in particular to detect fraudulent tax returns.



[http://as.wiley.com/WileyCDA/WileyTitle/productCd-1118152859.html]

Expectation

- The **expectation** (or **mean** or **expected value**) of a discrete random variable X is given by

$$\mathbb{E}X = \sum_x x p_X(x)$$

- To see why this makes sense, consider a RV X which takes only two possible values…

$$p_X(x) = \begin{cases} 1/3, & x = 3, \\ 2/3, & x = 4, \\ 0, & \text{otherwise.} \end{cases}$$

Analyze the following games (1)

Game #1

Flip a fair coin.

H: You get ฿100

T: You lose ฿100

Game #2

Flip a fair coin.

H: You get ฿200

T: You lose ฿100

Analyze the following games (2)

Game #3

Flip an *unfair* coin with $P(\{H\}) = 10^{-6}$

H: You get ฿2,000,000

T: You lose ฿0

Game #4

Pay ฿50 to play the game.

Flip an *unfair* coin with $P(\{H\}) = 10^{-6}$

H: You get ฿2,000,000

T: You lose ฿0

[http://www.glo.or.th] หวย (Huay)

Government Lottery (สลากกินแบ่งรัฐบาล)

เงื่อนไขเงินรางวัลสลากกินแบ่งรัฐบาล
(ใช้ตั้งแต่งวดวันที่ 1 พฤศจิกายน 2552 เป็นต้นไป)
สลาก 1 ชุด มี 1 ล้านฉบับๆ ละ 40 บาท
ถ้าจำหน่ายหมด กำหนดเงินรางวัลต่อชุด ดังนี้

รางวัลที่ หนึ่ง	มี	1 รางวัล ๆ ละ	2,000,000 บาท
รางวัลที่ สอง	มี	5 รางวัล ๆ ละ	100,000 บาท
รางวัลที่ สาม	มี	10 รางวัล ๆ ละ	40,000 บาท
รางวัลที่ สี่	มี	50 รางวัล ๆ ละ	20,000 บาท
รางวัลที่ ห้า	มี	100 รางวัล ๆ ละ	10,000 บาท
รางวัลข้างเคียงรางวัลที่หนึ่ง	มี	2 รางวัล ๆ ละ	50,000 บาท
รางวัลเลขท้าย 3 ตัว เสี่ยง 4 ครั้ง	มี	4,000 รางวัล ๆ ละ	2,000 บาท
รางวัลเลขท้าย 2 ตัว เสี่ยง 1 ครั้ง	มี	10,000 รางวัล ๆ ละ	1,000 บาท

สลาก 1 ชุด มี 14,168 รางวัล เป็นเงิน 23,000,000 บาท
รางวัลที่ 1 พิเศษ มี 2 รางวัล แบ่งเป็น 2 กลุ่ม
กลุ่มที่ 1 เท่ากับ จำนวนชุดที่ 01 ถึงชุดที่ 30 ที่จำหน่ายในแต่ละงวด x 1,000,000 บาท และยังมีสิทธิ์ได้รับเงินรางวัลอื่นอีก ตามเงื่อนไขเงินรางวัล
กลุ่มที่ 2 เท่ากับ จำนวนชุดที่ 51 ถึงชุดที่ 70 ที่จำหน่ายในแต่ละงวด x 1,000,000 บาท และยังมีสิทธิ์ได้รับเงินรางวัลอื่นอีก ตามเงื่อนไขเงินรางวัล

สำนักงานสลากกินแบ่งรัฐบาล ช่วยราษฎร์ เสริมรัฐ ยึดหยัดยุติธรรม

ผลการออกรางวัลสลากกินแบ่งรัฐบาล

งวดที่ 2 ประจำวันที่ 16 มกราคม พ.ศ. 2557

เป็นงวดที่ 17 สลากการกุศลงวดพิเศษ คณะแพทยศาสตร์โรงพยาบาลรามาธิบดี สลากการกุศลงวดพิเศษองค์การสงเคราะห์ทหารผ่านศึก สลากการกุศลงวดพิเศษมูลนิธิมิราเคิลออฟไลฟ์ สลากการกุศลงวดพิเศษโรงพยาบาลธรรมศาสตร์เฉลิมพระเกียรติ สลากการกุศลงวดพิเศษมูลนิธิวชิรพยาบาล สลากการกุศลงวดพิเศษโรงพยาบาลตำรวจ สลากการกุศลงวดพิเศษกระทรวงสาธารณสุข และเป็นงวดที่ 18 สลากการกุศลงวดพิเศษมูลนิธิอาสาเพื่อนพึ่ง (ภาฯ) ยามยาก

ดาวน์โหลด GLO Lottery ได้ที่ Google Play หรือ App Store	ตรวจผลรางวัลทันใจ หรือย้อนหลังได้ที่ โทร. 1900-1900-10 , 0-2528-8899	ตรวจผลทาง Internet www.glo.or.th

รางวัลที่ 1	เลขท้าย 3 ตัว				เลขท้าย 2 ตัว
สลากกินแบ่งรัฐบาล รางวัลละ 2,000,000 บาท สลากการกุศลงวดพิเศษ รางวัลละ 3,000,000 บาท	รางวัลละ 2,000 บาท				รางวัลละ 1,000 บาท
306902	**077**	**149**	**242**	**510**	**5 2**

รางวัลที่ 1 พิเศษ				
สลากกินแบ่งรัฐบาล กลุ่มที่ 1 รางวัลละ 30 ล้านบาท	ชุดที่	01	หมายเลข	306902
สลากกินแบ่งรัฐบาล กลุ่มที่ 2 รางวัลละ 20 ล้านบาท	ชุดที่	69	หมายเลข	306902

รางวัลข้างเคียงรางวัลที่ 1	รางวัลละ 50,000 บาท
3 0 6 9 0 1	3 0 6 9 0 3

รางวัลที่ 2 รางวัลละ 100,000 บาท				
160023	202375	416088	600455	945241

รางวัลที่ 3 รางวัลละ 40,000 บาท									
052424	253285	281902	292140	432317	721748	791326	831315	986137	987022

รางวัลที่ 4 รางวัลละ 20,000 บาท									
003040	015566	061042	216269	282188	350069	479231	671571	750280	946494
005150	025363	066647	236669	290420	388294	498251	694605	783280	948660
011032	042073	087239	258907	298872	393732	597084	699036	817745	962443
011724	044016	105036	261336	337144	427014	617406	715866	878761	977353
014755	053924	202281	269387	339614	439462	626098	730109	899240	995117

รางวัลที่ 5 รางวัลละ 10,000 บาท									
002000	121588	248663	344930	402741	544190	642420	721226	804484	856226
018528	128288	260429	350947	403009	555663	652033	734375	804867	880497
025513	136945	314767	351906	411908	555825	660690	743264	813393	882449
043369	154409	314880	361460	416269	566821	677460	747732	830580	883583
067776	154939	325705	372413	417017	567242	684084	749215	831530	892477
079139	194574	328436	375163	441655	572170	691506	765807	838709	917135
095994	200129	328632	375630	485506	572322	702562	772720	842864	939551
102279	220150	334961	376842	488679	608519	708438	786756	844814	949966
104520	243342	335959	396594	503022	619829	719216	786850	850780	965838
110065	244491	343245	399864	512197	620249	720699	788260	855266	985581

Government Lottery (สลากกินแบ่งรัฐบาล)

ตารางที่ 4 การคำนวณกำไรคาดหวังของสลากกินแบ่งรัฐบาล

ชื่อรางวัล	จำนวนรางวัล	กำไร(1) =เงินรางวัล-ค่าซื้อสลาก[1]	โอกาสที่จะถูกรางวัล (2)
รางวัลที่ 1 ชุดใหญ่ 30 ล้านบาท	1	30 ล้าน - 40 บาท	0.00000333 %
รางวัลที่ 1 ชุดใหญ่ 16 ล้านบาท	1	16 ล้าน - 40 บาท	0.00000625 %
รางวัลที่ 1	46	2 ล้าน -40 บาท	0.0001%
รางวัลข้างเคียงรางวัลที่ 1	92	5 หมื่น -40 บาท	0.0002%
รางวัลที่ 2	230	1 แสน -40 บาท	0.0005%
รางวัลที่ 3	460	4 หมื่น -40 บาท	0.001%
รางวัลที่ 4	2,300	2 หมื่น -40 บาท	0.005%
รางวัลที่ 5	4,600	1 หมื่น -40 บาท	0.01%
เลขท้าย 3 ตัว	184,000	2 พัน -40 บาท	0.4%
เลขท้าย 2 ตัว	460,000	1 พัน -40 บาท	1.0%
สลากที่ไม่ถูกรางวัลใดๆ	-	-40 บาท	98.58%

[1] ไม่ได้รวมภาษีหัก ณ ที่จ่ายไม่เกินร้อยละ 1 ของเงินรางวัล เพื่อง่ายต่อการคำนวณ

$$\text{Expected Profit} = -16$$

Can only press once

- Can only press once – Instant $1 Million or 50% chance for $100 million

Expectation and Variance

- The **expectation** (or **mean** or **expected value**) of a discrete random variable X is given by

$$\mathbb{E}X = \sum_x x p_X(x)$$

- The expected value of a function g of a RV X is given by

$$\mathbb{E}[g(X)] = \sum_x g(x) p_X(x)$$

- The **variance** of a RV X is given by

$$\mathrm{Var}[X] = \mathbb{E}\left[(X - \mathbb{E}X)^2\right] = \mathbb{E}[X^2] - (\mathbb{E}X)^2$$

- The **standard deviation** of a RV X is given by

$$\sigma_X = \sqrt{\mathrm{Var}[X]}$$

Continuous Random Variables

- Recall: X is a **discrete** random variable if it has a countable support.
- X is a **continuous** random variable if we can find a function f such that

$$P[a \le X \le b] = \int_a^b f(x)dx$$

  - The function f is called the **probability density function** (**pdf**) or simply **density**.
  - When we want to emphasize that the function f is a density of a particular random variable X, we write f_X instead of f.

Examples

- For the random variable X generated by `X = rand` in MATLAB,

$$f_X(x) = \begin{cases} 1, & 0 \le x \le 1, \\ 0, & \text{otherwise.} \end{cases}$$

- For the random variable X generated by `X = randn` in MATLAB,

$$f_X(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2}$$

Expectation and Variance

- The **expectation** (or **mean** or **expected value**) of a continuous random variable X is given by

$$\mathbb{E}X = \int_{-\infty}^{\infty} x f_X(x) dx$$

- The expected value of a function g of a RV X is given by

$$\mathbb{E}\left[g(X)\right] = \int_{-\infty}^{\infty} g(x) f_X(x) dx$$

- The **variance** of a RV X is given by

$$\mathrm{Var}[X] = \mathbb{E}\left[(X - \mathbb{E}X)^2\right] = \mathbb{E}\left[X^2\right] - (\mathbb{E}X)^2$$

- The **standard deviation** of a RV X is given by

$$\sigma_X = \sqrt{\mathrm{Var}[X]}$$

Symbolic Computations in MATLAB

- Symbolic Math Toolbox
  - The Symbolic Math Toolbox is included in the Student Version of MATLAB.
- Functions for computing, solving, and manipulating symbolic math expressions and performing variable-precision arithmetic.
- Can analytically perform
  - Differentiation (including partial differentiation)
  - (Definite and indefinite) integration
  - Taking limits (including one-sided limits)
  - Summation (including Taylor series)
  - Simplification
  - Matrix operations
  - (Integral) transforms (including Fourier, Laplace, Z)
  - (Algebraic and differential) equation solving
- Data type: symbolic objects
  - symbolic variables, symbolic numbers, symbolic expressions, symbolic matrices, and symbolic functions.

Symbolic Variables

- Use **`sym`** or **`syms`** to create symbolic variables.
  - The syms command:
    - Does not use parentheses and quotation marks:
      `syms x`
    - Can create multiple objects with one call:
      `syms x y z`
  - The sym command:
    - Requires parentheses and quotation marks:
      `x = sym('x')`.
    - Creates one symbolic object with each call.
- Can manipulate the symbolic objects according to the usual rules of mathematics.

```
>> syms x y z
>> A = [x y; z x]

A =

[ x, y]
[ z, x]

>> sum(A)

ans =

[ x + z, x + y]

>> sum(A,2)

ans =

 x + y
 x + z
```

Symbolic Numbers

- To convert a number to a symbolic number, use the `sym` command
  - `x = sym('2')`
- If you create a symbolic number with 15 or fewer decimal digits, you can skip the quotes:
  - `x = sym(2)`
- You also can create a rational fraction involving symbolic numbers:
  - `x = sym(2)/sym(5)`
  - `x = sym(2/5)`
- To evaluate a symbolic number numerically, use the **`double`** command:
  - `double(x)`

```
>> x = sym(2/5)

x =

2/5

>> double(x)

ans =

    0.4000
```

Double-precision vs symbolic number

- By default, the `sym` command returns a rational *approximation* of a numeric expression.
- Symbolic results are not indented.
- Standard MATLAB double-precision results are indented.

```
>> x = 0.25

x =

    0.2500

>> x = sym(x)

x =

1/4
```

```
>> sym(sqrt(2)+1)

ans =

2718162824974067/1125899906842624

>> sym(sqrt(sym(2))+1)

ans =

2^(1/2) + 1
```

Double-precision vs symbolic number

- If you want to ensure a precise symbolic expression, you must avoid numeric computations.
- Compare these three expressions.
  - The first is only accurate to double-precision numeric computation (about 16 digits).
  - The second and third avoid numeric computation completely.

```
>> sym(log(2))

ans =

6243314768165359/9007199254740992

>> sym('log(2)')

ans =

log(2)

>> log(sym(2))

ans =

log(2)
```

Symbolic Expressions

- `x = sym('(sqrt(2)+1)/3')`
- `b = sym('a^2+1')`
- Note that the second statement above does not create (symbolic) variable `a`.

Same name

- Many of the functions in the Symbolic Math Toolbox have the same names as their numeric counterparts.
  - MATLAB selects the correct one depending on the type of inputs to the function.
- Example:
  - **`diff`** calculates differences between adjacent elements (which can be used to numerically approximate the derivative of a function)
    - `help diff`,
    - `doc diff`
  - `symbolic/`**`diff`** differentiates symbolic expression
    - `help sym/diff`
    - `doc symbolic/diff`

```
>> x = [1 4 6]

x =

     1     4     6

>> diff(x)

ans =

     3     2

>> x = sym('x')

x =

x

>> diff(x)

ans =

1
```

Symbolic Functions

- `syms f(x,y)` creates the symbolic function `f` and symbolic variables `x` and `y`.
- Alternatively, you can use `sym` to create a symbolic function.
  - Note that sym only creates the function. It does not create symbolic variables that represent its arguments. You must create these variables before creating a function:
  - `syms x y;`
  - `f(x,y) = sym('f(x,y)');`
- Create a function defined by a particular mathematical expression
  - `syms x y`
  - `f(x,y) = x^3*y^3`
- After creating a symbolic function, you can differentiate, integrate, or simplify it, substitute its arguments with values, and perform other mathematical operations

```
>> syms x y
>> f(x,y) = x^3*y^3

f(x, y) =

x^3*y^3

>> f(1,3)

ans =

27

>> f([1 2],[3 4])

ans =

[ 27, 512]
```

```
>> diff(f,x)

ans(x, y) =

3*x^2*y^3
```

Calculus: `diff`

- `diff(S)` differentiates a symbolic expression `S`.
  - If you do not specify any variable, MATLAB chooses a **default variable** by the proximity to the letter `x`.
- `diff(S,'v')` or `diff(S,sym('v'))` differentiates `S` with respect to `v`.
  - Can find partial derivative
- The `diff` function can also take a symbolic matrix as its input. In this case, the differentiation is done element-by-element.

```
>> syms x
>> f = x^2 * exp(x)

f =

x^2*exp(x)

>> diff(f)

ans =

x^2*exp(x) + 2*x*exp(x)
```

differentiates `f` with respect to `x`

Derivatives of Expressions with Several Variables

```
>> syms x y
>> f = x^2*y^3

f =

x^2*y^3

>> diff(f,x)

ans =

2*x*y^3

>> diff(f,y)

ans =

3*x^2*y^2
```

$\frac{\partial f}{\partial x}$

$\frac{\partial f}{\partial y}$

Default Symbolic Variable

- If you do not specify an independent variable when performing substitution, differentiation, or integration, MATLAB uses a **default variable**.
- The default variable is typically the one closest alphabetically to `x` or, for symbolic functions, the first input argument of a function.
- To determine the default variable, use **`symvar`**.

```
>> syms s t
>> f = s*t

f =

s*t

>> symvar(f,1)

ans =

t

>> diff(f)

ans =

s

>> diff(f,t)

ans =

s

>> diff(f,s)

ans =

t
```

The letter `t` is closer to x in the alphabet than the letter `s` is.

Calculus: `diff`

- `diff(S,n)`, for a positive integer `n`, differentiates `S` `n` times.
- `diff(S,'v',n)` and `diff(S,n,'v')` are also acceptable.

```
>> syms x
>> f = x^3

f =

x^3

>> diff(f)

ans =

3*x^2
```

```
>> diff(diff(f))

ans =

6*x

>> diff(f,2)

ans =

6*x
```

Interesting Example: Abstract functions

```
>> syms x n
>> f = sym('f(x)')

f =

f(x)

>> g = sym('g(x)')

g =

g(x)

>> diff(f*g)

ans =

f(x)*diff(g(x), x) + g(x)*diff(f(x), x)

>> diff(f^n)

ans =

n*f(x)^(n - 1)*diff(f(x), x)
```

Calculus: `int`

- `int(S)` is the indefinite integral of `S`.
- `int(S,v)` is the indefinite integral of `S` with respect to `v`.
- `int(S,a,b)` is the definite integral of `S` from `a` to `b`.
  - `a` and `b` are each double or symbolic scalars.
    - `inf` is also OK.
- `int(S,v,a,b)` is the definite integral of `S` with respect to `v` from `a` to `b`.

```
>> int(exp(-x^2), -inf, inf)

ans =

pi^(1/2)
```

```
>> syms x
>> f = x^2*exp(x)

f =

x^2*exp(x)

>> int(f)

ans =

exp(x)*(x^2 - 2*x + 2)

>> int(f,0,2)

ans =

2*exp(2) - 2

>> syms a
>> int(f,0,a)

ans =

exp(a)*(a^2 - 2*a + 2) - 2
```

```
>> syms x y
>> f = x^2*y^3

f =

x^2*y^3

>> int(f,x)

ans =

(x^3*y^3)/3

>> int(f,y)

ans =

(x^2*y^4)/4

>> int(f,y,0,2)

ans =

4*x^2
```

Assumptions on Symbolic Objects

```
>> syms x n
>> f = x^n;
>> int(f)

ans =

piecewise([n == -1, log(x)], [n ~= -1, x^(n + 1)/(n + 1)])
```

$$\int x^n dx = \begin{cases} \ln(x), & n = 1, \\ \dfrac{x^{n+1}}{n+1}, & n \neq 1. \end{cases}$$

```
>> assume(n ~= -1)
>> int(f)

ans =

x^(n + 1)/(n + 1)
```

```
>> syms x a
>> f = exp(-a*x^2);
>> int(f, x, -inf, inf)

ans =

piecewise([a < 0, Inf], [0 <= real(a) or (angle(a) ir

>> assume(a > 0)
>> int(f, x, -inf, inf)

ans =

pi^(1/2)/a^(1/2)
```

$$\int_{-\infty}^{\infty} e^{-ax^2} dx = \sqrt{\frac{\pi}{a}}$$

Assumptions on Symbolic Objects

- Symbolic variables are complex variables by default.
- To **set** an assumption on a symbolic variable, use the **`assume`** function.
  - Assume replaces all previous assumptions on the variable with the new assumption.
  - For example, assume that the variable `x` is nonnegative:
    - `syms x`
    - `assume(x >= 0)`
- If you want to add a new assumption to the existing assumptions, use **`assumeAlso`**.
  - For example, add the assumption that `x` is also an integer.
    - `assumeAlso(x,'integer')`
    - Now the variable x is a nonnegative integer:

Assumptions on Symbolic Objects

- `assume` and `assumeAlso` let you state that a variable or an expression belongs to one of these sets:
  - integers, rational numbers, and real numbers.
- Alternatively, you can set an assumption while declaring a symbolic variable by the `sym` or `syms` command:
  - Two assignable assumptions: **`real`** and **`positive`**.

```
>> a = sym('a','real');
>> b = sym('b','real');
>> c = sym('c','positive');
```

```
syms a b real
syms c positive
```

- To check existing assumptions,

```
>> assumptions

ans =

[ a in R_, b in R_, 0 < c]
```

```
>> assumptions(c)

ans =

0 < c
```

Deleting Symbolic Objects and Their Assumptions

- Symbolic objects and their assumptions are stored separately.
  - The object is stored in the MATLAB workspace, and the assumption is stored in the symbolic engine.
- When you delete a symbolic object from the MATLAB workspace using `clear x`, the assumption of `x` still remains in the symbolic engine.
  - If you declare a new symbolic variable `x` later, it inherits the old assumption instead of getting a default assumption.
- If you want to remove both the symbolic object and its assumption, use two subsequent commands:
  - `syms x clear`
    - clear the assumption,
  - `clear x;`
    - delete the symbolic object

Calculus: `limit`

- `limit(expr,x,x0)` computes limit of the symbolic expression when `x` approaches `x0`.
- `limit(expr,c)` computes limit of the symbolic expression when the default variable approaches `c`.
- `limit(expr)` computes limit of the symbolic expression when the default variable approaches 0.

```
>> syms x c
>> limit(sin(x)/x)

ans =

1

>> limit((1+c/x)^x,x,inf)

ans =

exp(c)
```

$$\lim_{x\to 0}\frac{\sin x}{x}=1$$

$$\lim_{x\to\infty}\left(1+\frac{c}{x}\right)^{x}=e^{c}$$

Calculus: One-Sided Limits

- `limit(expr,x,x0,'left')` computes the limit of the symbolic expression when x approaches `x0` from the left.
- `limit(expr,x,x0,'right')` computes the limit of the symbolic expression when x approaches `x0` from the right.

Since the limit from the left does not equal the limit from the right, the two-sided limit does not exist. In the case of undefined limits, MATLAB returns **NaN** (not a number).

```
>> syms x
>> f = x/abs(x)

f =

x/abs(x)

>> limit(f,x,0,'left')

ans =

-1
```

$$\boxed{\lim_{x\to 0^-}\frac{x}{|x|}=-1}$$

```
>> limit(f,x,0,'right')

ans =

1
```

$$\boxed{\lim_{x\to 0^+}\frac{x}{|x|}=1}$$

```
>> limit(x/abs(x), x, 0)

ans =

NaN
```

Calculus: `limit`

```
>> syms x
>> f(x) = sin(x)

f(x) =

sin(x)

>> syms h
>> limit((f(x+h)-f(x))/h,h,0)

ans =

cos(x)
```

Recall, from calculus, that

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

Also recall that

$$\frac{d}{dx}\sin(x) = \cos(x)$$

Summary

Mathematical Operation	MATLAB Command
$\lim_{x \to 0} f(x)$	`limit(f)`
$\lim_{x \to a} f(x)$	`limit(f, x, a)` or `limit(f, a)`
$\lim_{x \to a^-} f(x)$	`limit(f, x, a, 'left')`
$\lim_{x \to a^+} f(x)$	`limit(f, x, a, 'right')`

Mathematical Operator	MATLAB Command
$\frac{df}{dx}$	`diff(f)` or `diff(f, x)`
$\frac{df}{da}$	`diff(f, a)`

Mathematical Operator	MATLAB Command
$\frac{d^2 f}{db^2}$	`diff(f, b, 2)`
$J = \frac{\partial(r,t)}{\partial(u,v)}$	`J = jacobian([r; t],[u; v])`

Definite Integral	Command
$\int_a^b f(x)dx$	`int(f, a, b)`
$\int_a^b f(v)dv$	`int(f, v, a, b)`

pretty

- Print/display symbolic output in an “easy-to-read” form resembling typeset mathematics.

```
>>  A = [sym(3/2) - 5^(1/sym(2))/2; sym(3/2) + 5^(1/sym(2))/2]

A =

 3/2 - 5^(1/2)/2
 5^(1/2)/2 + 3/2

>> pretty(A)

  +-             -+
  |         1/2   |
  |        5      |
  |  3/2 - ----   |
  |         2     |
  |               |
  |   1/2         |
  |  5            |
  |  ---- + 3/2   |
  |   2           |
  +-             -+
```

vpa

- **vpa** = **Variable precision arithmetic**
- Numeric computations in MATLAB are done in approximately 16 decimal digit floating-point arithmetic.
- With vpa, you can obtain results to arbitrary precision, within the limitations of time and memory.
  - The default precision for vpa is 32.
- Caution: If you pass a numeric expression to vpa, MATLAB evaluates it numerically first.
  - So use a symbolic expression or place the expression in quotes.
- Examples
  - The first results are accurate to approximately 16 digits
  - The next two results are accurate to 32 digits
  - The third result is accurate to the specified 50 digits.
  - The 4th result is accurate to only about 16 digits (even though 50 digits are displayed).

```
>> format long
>> pi*log(2)

ans =

   2.177586090303602

>> vpa('pi*log(2)')

ans =

2.1775860903036021305006888982376

>> vpa(sym(pi)*log(sym(2)))

ans =

2.1775860903036021305006888982376

>> vpa('pi*log(2)',50)

ans =

2.1775860903036021305006888982376139473385837003693
>> vpa(pi*log(2),50)

ans =

2.1775860903036021731793425715295597910881042480469
```

Substitution

- The function **subs** replaces all occurrences of the symbolic variable in an expression by a specified second expression.

```
>> syms x
>> subs(sin(x),x,pi/3)

ans =

3^(1/2)/2
```

```
>> syms w t
>> x(t) = sin(2*pi*w*t)

x(t) =

sin(2*pi*t*w)

>> x(1)

ans =

sin(2*pi*w)

>> subs(x,w,5)

ans(t) =

sin(10*pi*t)

>> x(1)

ans =

sin(2*pi*w)
```

```
>> x = subs(x,w,5)

x(t) =

sin(10*pi*t)

>> x(1)

ans =

0
>> x = subs(x,t,cos(t))

x(t) =

sin(10*pi*cos(t))

>> x = subs(x,10*cos(t),t)

x(t) =

sin(pi*t)
```

Substitution

- You can substitute multiple symbolic expressions, numeric expressions, or any combination, using cell arrays of symbolic or numeric values.

Interesting Example: Abstract functions

```
>> f = sym('f(x)');
>> g = sym('g(x)');
>> diff(subs(f, g))

ans =

D(f)(g(x))*diff(g(x), x)
```

```
>> syms x y
>> S = x^y

S =

x^y

>> subs(S, x, 3)

ans =

3^y

>> subs(S, {x y}, {3 2})

ans =

9

>> subs(S, {x y}, {y x})

ans =

y^x

>> subs(S, x, 1:3)

ans =

[ 1, 2^y, 3^y]

>> subs(S, {x y}, {1:3 -1:1})

ans =

[ 1, 1, 3]
```

Algebraic simplification

- **expand**(S) expands the symbolic expression S.
  - Most often used on polynomials (distributing products over sums, multiplying out terms), but also expands trigonometric, exponential and logarithmic functions.
- **factor**(S) factorizes the symbolic expression S.
  - If S contains all integer elements, the prime factorization is computed.
- **collect**(S) views a symbolic expression as a polynomial in its symbolic variable (which may be specified) and collects all terms with the same power of the variable.

```
>> f = (3*x+x*y)^3

f =

(3*x + x*y)^3

>> expand(f)

ans =

x^3*y^3 + 9*x^3*y^2 + 27*x^3*y + 27*x^3

>> factor(f)

ans =

x^3*(y + 3)^3

>> collect(f,x)

ans =

(y + 3)^3*x^3

>> collect(f,y)

ans =

x^3*y^3 + 9*x^3*y^2 + 27*x^3*y + 27*x^3
```

`simplify` and `simple`

- Function `simplify` applies many identities in an attempt to reduce a symbolic expression to a simple form.
  - You can also use the syntax `simplify(f,'Steps',n)` where `n` is a positive integer that controls how many steps simplify takes.
    - By default, `n = 1`.

```
>> syms x
>> z = (cos(x)^2 - sin(x)^2)*sin(2*x)*(exp(2*x) - 2*exp(x) + 1)/(exp(2*x) - 1);
>> simplify(z)

ans =

(sin(4*x)*(exp(x) - 1))/(2*(exp(x) + 1))

>> simplify(z, 'Steps', 30)

ans =

(sin(4*x)*tanh(x/2))/2
```

- The alternate function `simple` computes several simplifications and chooses the shortest of them.

symsum: Symbolic Summation

```
>> syms x k
>> s1 = symsum(1/k^2, 1, inf)

s1 =

pi^2/6

>> s2 = symsum(x^k, k, 0, inf)

s2 =

piecewise([1 <= x, Inf], [abs(x) < 1, -1/(x - 1)])
```

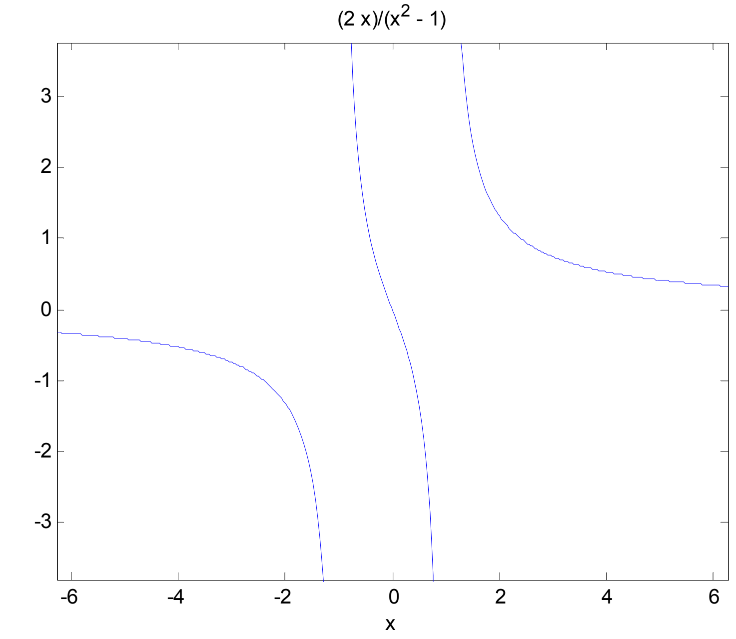
$$\sum_{k=1}^{\infty} \frac{1}{k^2} = 1 + \frac{1}{2^2} + \frac{1}{3^2} + \cdots = \frac{\pi^2}{6}$$

$$\sum_{k=0}^{\infty} x^k = 1 + x + x^2 + \cdots = \begin{cases} \dfrac{1}{1-x}, & |x| < 1, \\ \infty, & |x| \geq 1. \end{cases}$$

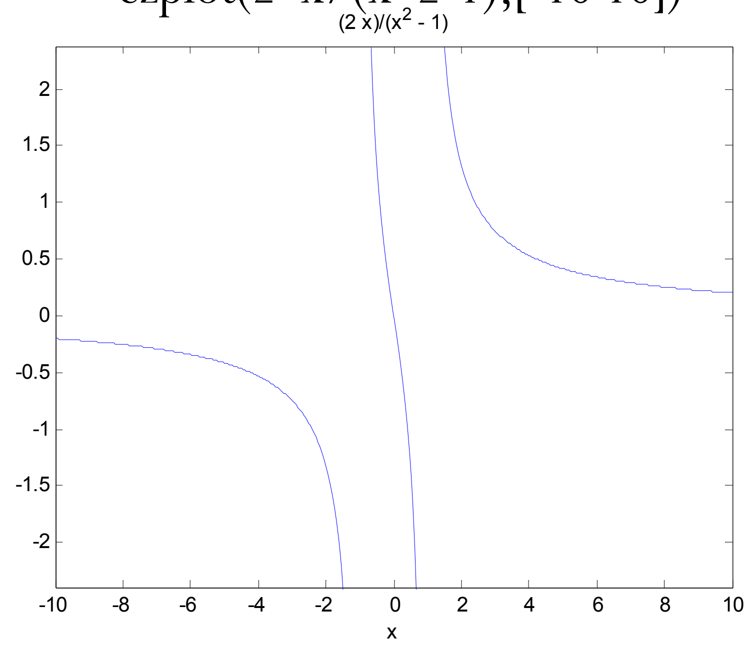
Plot

- There are several plot functions in MATLAB with names beginning with “ez” that perform the necessary conversions from symbolic expressions to numbers and plot them.
- **`ezplot`** lets you plot the graph of a function directly from its defining symbolic expression.
- By default, the `x`-domain is [-2π, 2 π].
  - This can be overridden by a second input variable.

syms x; ezplot(2*x/(x^2-1))



ezplot(2*x/(x^2-1),[-10 10])



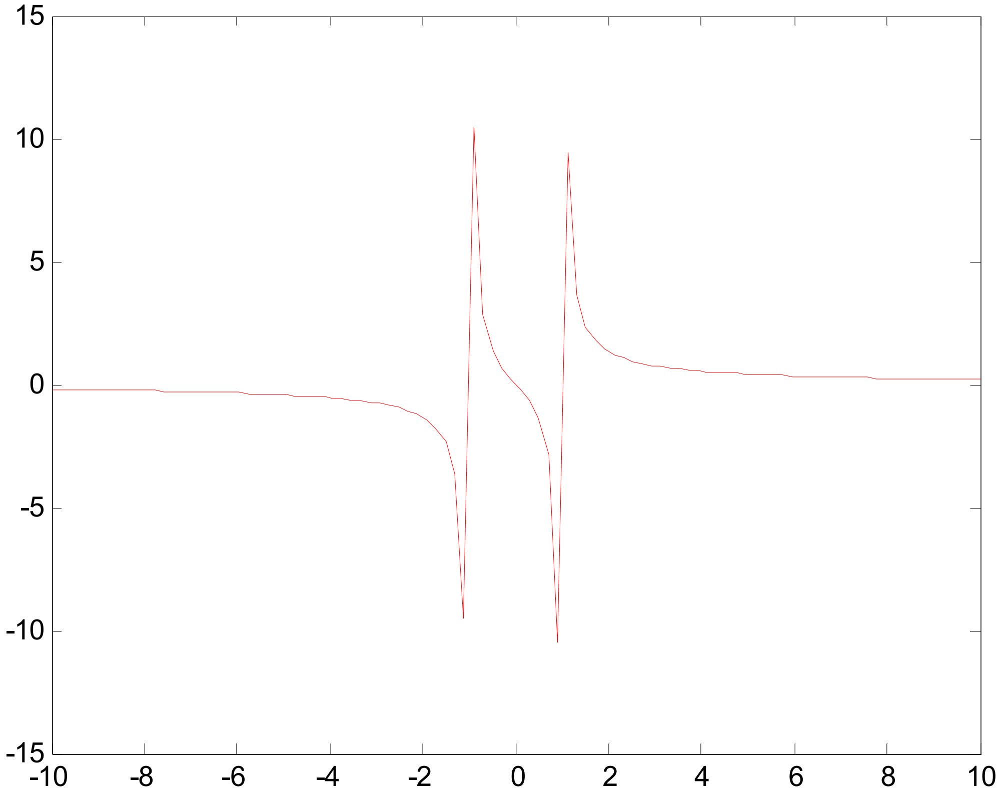
Use `subs` and `double` for more control in plotting

```
>> syms x
>> f = 2*x/(x^2-1)

f =

(2*x)/(x^2 - 1)

>> X = linspace(-10,10,100);
>> plot(X,double(subs(f,x,X)),'r')
```



Solve: Solving algebraic equations

- The inputs to `solve` can be quoted strings or symbolic expressions.

Use the double equal sign (`==`) to define an equation

```
>> syms x
>> solve(x^3 - 6*x^2 == 6 - 11*x)

ans =

 1
 2
 3

>> solve(x^3 - 6*x^2 - 6 + 11*x)

ans =

 1
 2
 3
```

```
>> solve('x^3 - 6*x^2 = 6 - 11*x')

ans =

 1
 2
 3

>> solve('x^3 - 6*x^2 - 6 + 11*x')

ans =

 1
 2
 3
```

If you do not specify the right side of the equation, `solve` assumes that it is zero

Solving algebraic equations

- The solve function cannot solve all equations. It does well with low-degree polynomial equations, but can have difficulty with trigonometric or other transcendental equations.
- If an exact symbolic solution is found, you can convert it to a floating-point solution via double.
- If an exact symbolic solution cannot be found, then a variable precision one is computed.

```
>> syms x b
>> solve(2^x - b)

ans =

log(b)/log(2)

>> solve(2^x + 3^x - 1)

ans =

-0.78788491102586978362855591729843

>> solve(2^x + 3^x - b)
Warning: Explicit solution could not be
found.
> In solve at 179

ans =

[ empty sym ]
```

Solving algebraic equations

```
>> x = solve('log(x) = x-2')

x =

-lambertw(0, -exp(-2))

>> double(x)

ans =

   0.158594339563039

>> vpa(x)

ans =

0.15859433956303936215339534198751
>> solve('x-3')

ans =

3

>> x
Undefined function or variable 'x'.
```

```
>> solve('1 + (a+b)/(a-b) = b', 'a')

ans =

b^2/(b - 2)

>> b
Undefined function or variable 'b'.
```

```
>> clear all
>> a = solve('1 + (a+b)/(a-b) = b', 'a')

a =

b^2/(b - 2)

>> subs(a,'b',1)

ans =

-1

>> subs(a,b,1)
Undefined function or variable 'b'.
```

Solving algebraic equations

```
>> syms a b c x
solve(a*x^2 + b*x + c, x)
pretty(ans)

ans =

 -(b + (b^2 - 4*a*c)^(1/2))/(2*a)
 -(b - (b^2 - 4*a*c)^(1/2))/(2*a)


  +-                        -+
  |         2          1/2   |
  |    b + (b   - 4 a c)     |
  |  - -------------------   |
  |           2 a            |
  |                          |
  |         2          1/2   |
  |    b - (b   - 4 a c)     |
  |  - -------------------   |
  |           2 a            |
  +-                        -+
```

This is a symbolic vector whose elements are the two solutions.

Solving Systems of Algebraic Equations

- The function solve can also compute solutions of systems of general algebraic equations

```
>> S1 = 'x^2 + y^2 + z^2 = 2'

S1 =

x^2 + y^2 + z^2 = 2

>> S2 = 'x + y = 1'

S2 =

x + y = 1

>> S3 = 'y + z = 1'

S3 =

y + z = 1
```

```
>> syms x y z
>> S1 = x^2 + y^2 + z^2 == 2

S1 =

x^2 + y^2 + z^2 == 2

>> S2 = x+y == 1

S2 =

x + y == 1

>> S3 = y + z == 1

S3 =

y + z == 1
```

```
>> [X, Y, Z] = solve(S1, S2, S3)

X =

    1
 -1/3


Y =

   0
 4/3


Z =

    1
 -1/3
```

Solving differential equations

First-Order ODE

```
>> syms y(t)
>> y(t) = dsolve(diff(y) == t*y)

y(t) =

C2*exp(t^2/2)
```

```
>> syms y(t)
>> y(t) = dsolve(diff(y) == t*y, y(0) == 2)

y(t) =

2*exp(t^2/2)
```

Second-Order ODE

```
>> syms y(x)
>> Dy = diff(y);
>> y(x) = dsolve(diff(y, 2) == cos(2*x) - y, y(0) == 1, Dy(0) == 0);
>> y(x) = simplify(y)

y(x) =

1 - (8*sin(x/2)^4)/3
```

Solving differential equations

- The function **dsolve** solves ordinary differential equations.
- The symbolic differential operator is `D`.
- If no independent variable is supplied, then it is assumed to be `t`.
- The higher order symbolic differential operators `D2`, `D3`, … can be used to solve higher order equations.

```
>> Y = dsolve('Dy = x^2*y','x')

Y =

C4*exp(x^3/3)

>> Y = dsolve('Dy = x^2*y', 'y(0)=4', 'x')

Y =

4*exp(x^3/3)

>> Y = dsolve('Dy = x^2*y')

Y =

C2*exp(t*x^2)
```