Matlab Programming Tips

2.01.20140607, 1.02.20090521

魏为民
afoe@163.com

上 海 电 力 学 院
计算机科学与技术学院

Syllabus

- 软件工程
- 匈牙利命名法
- 温伯格！布鲁克斯！克努斯！
- **Matlab Programming Tips and Tricks**
- **FAQs**
- 在MATLAB7.0中编译和发布可执行文档
- 总结

软件工程

- 计算机系统＝软件＋硬件＋数据库＋人＋过程等
- 软件＝程序＋数据＋文档＋Demo
- 软件的特点
  - 抽象性
  - 软件生产无明显制造过程
  - 无磨损性
  - 对计算机硬件依赖性
  - 软件的手工开发方式
  - 软件本身复杂性
  - 软件的高成本

软件工程的发展

- 软件的发展经历了三个阶段
  - 程序设计阶段 — 50至60年代
  - 程序系统阶段 — 60至70年代
  - 软件工程阶段 — 70年代以后
- 软件工程概念的出现源自软件危机。

软件工程的定义

- 软件工程：运用现代科学技术知识来设计并构造计算机程序及为开发、运行和维护这些程序所必须的相关文档资料。
- 软件工程学：建立并使用完善的工程化原则，以较经济的手段获得能在实际机器上有效运行的可靠软件的一系列方法。
- 软件工程三要素：方法、工具和过程。

软件工程的目标

- 软件工程需要解决的主要问题：软件成本、软件可靠性、软件维护、软件生产率和软件复用。
- 软件工程需要达到的基本目标：
  - 付出较低的开发成本
  - 达到要求的软件功能
  - 取得较好的软件性能
  - 开发的软件易于移植
  - 需要较低的维护费用
  - 能按时完成开发，及时交付使用

软件工程

- 软件过程：是把输入转化为输出的一组彼此相关的资源和活动。
- 从软件开发的观点看，它就是使用适当的资源（包括人员、硬软件工具、时间等），为开发软件进行的一组开发活动，在过程结束时将输入（用户要求）转化为输出（软件产品）。

软件工程

- 软件过程定义了：方法使用的顺序、 要求交付的文档资料、为保证质量和适应变化所需要的管理、软件开发各个阶段完成的里程碑。
- 四种基本的过程活动：
  - **Plan** 软件规格说明
  - **Do** 软件开发
  - **Check** 软件确认
  - **Action** 软件演进

软件工程

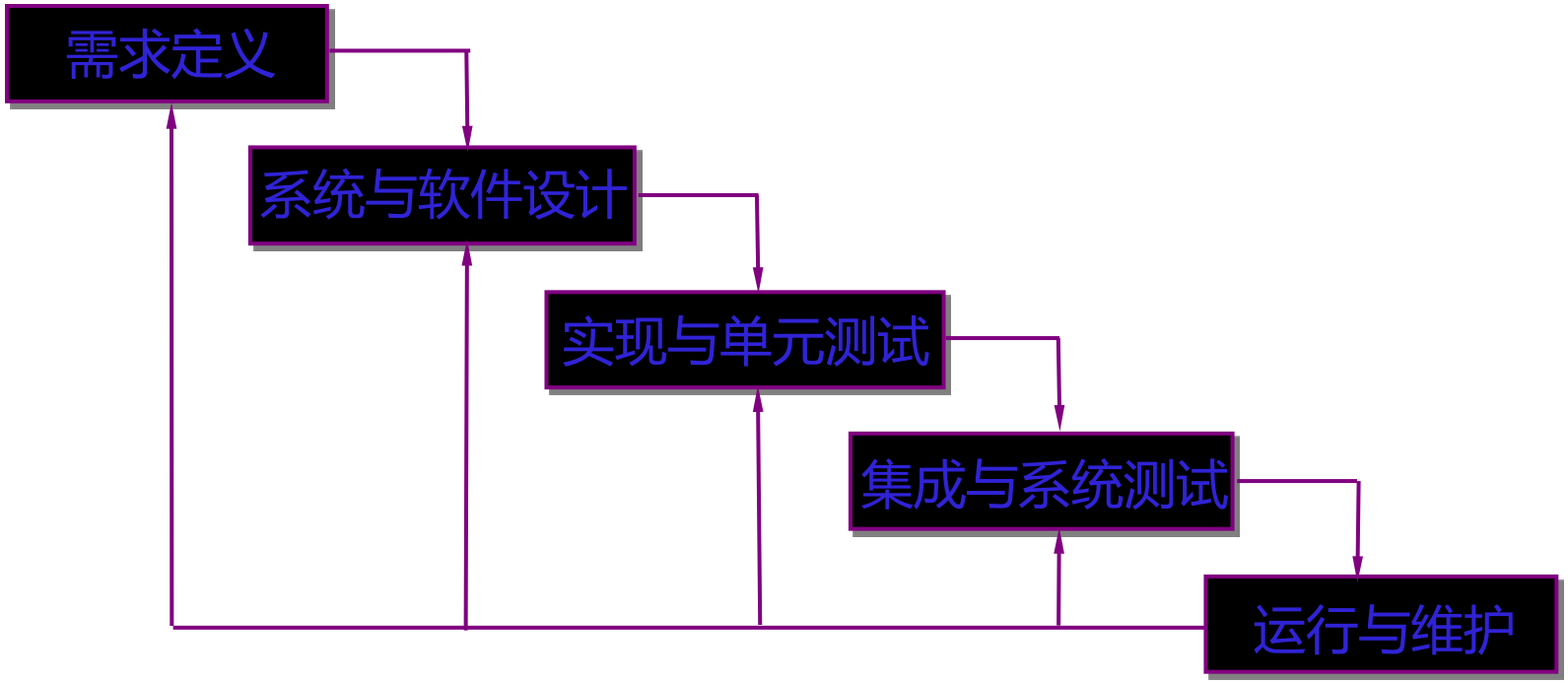
- 软件有一个孕育、诞生、成长、成熟、衰亡的生存过程。这个过程即为软件的生存期。
- 软件生存期包含三个阶段：软件定义、软件开发及软件运行维护。
- 软件生存期模型是软件工程思想的具体化，是跨越软件生存期的系统开发、运行、维护所实施的全部活动和任务的过程框架。

软件生存期模型

- 常用的软件生存期模型有：
  - 瀑布模型
  - 演化模型
  - 螺旋模型
  - 增量模型
  - 喷泉模型
  - 智能模型

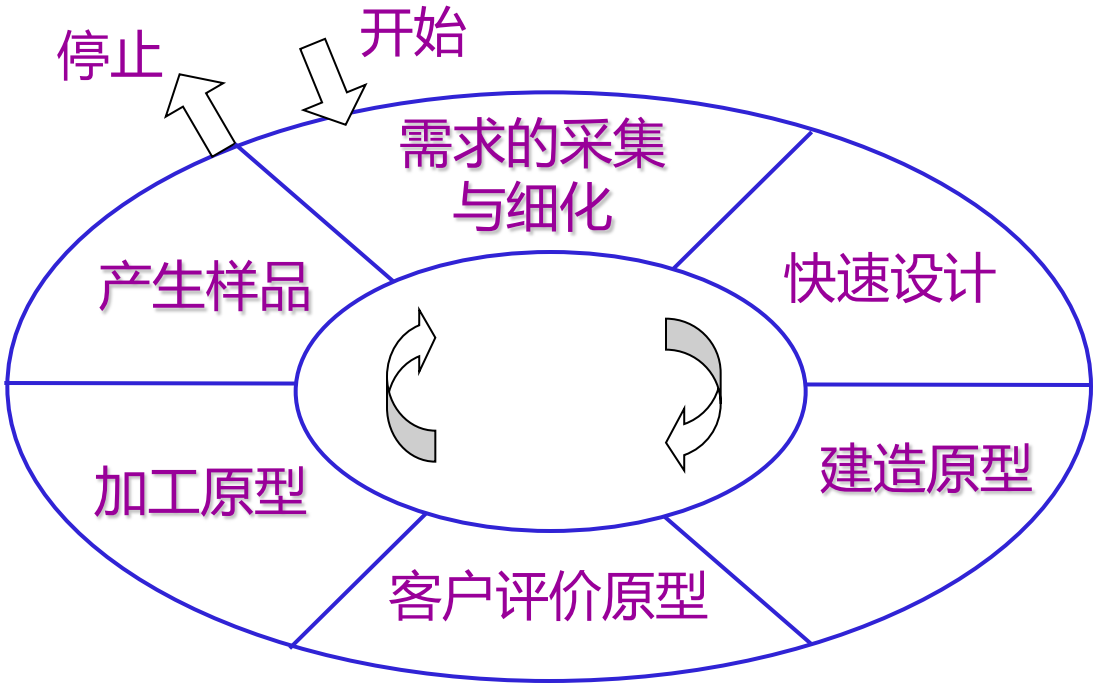
软件工程：瀑布模型

- 各项活动按自上而下，相互衔接的固定次序，如同瀑布逐级下落。每项活动均处于一个质量环（输入-处理-输出-评审）中。



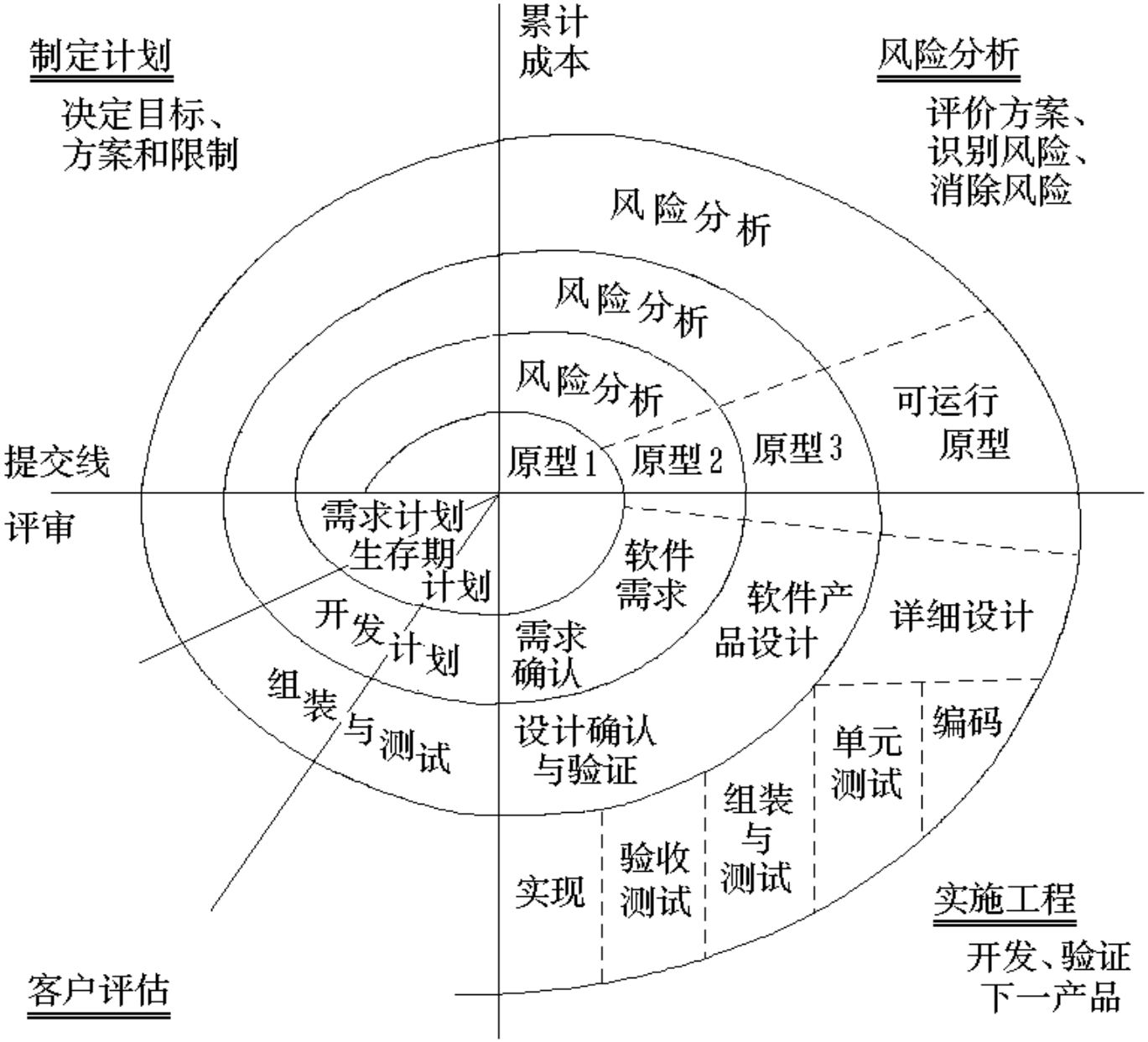
软件工程：演化模型

- 先开发一个“原型”软件，完成部分主要功能，展示给用户并征求意见，然后逐步完善，最终获得满意的软件产品。



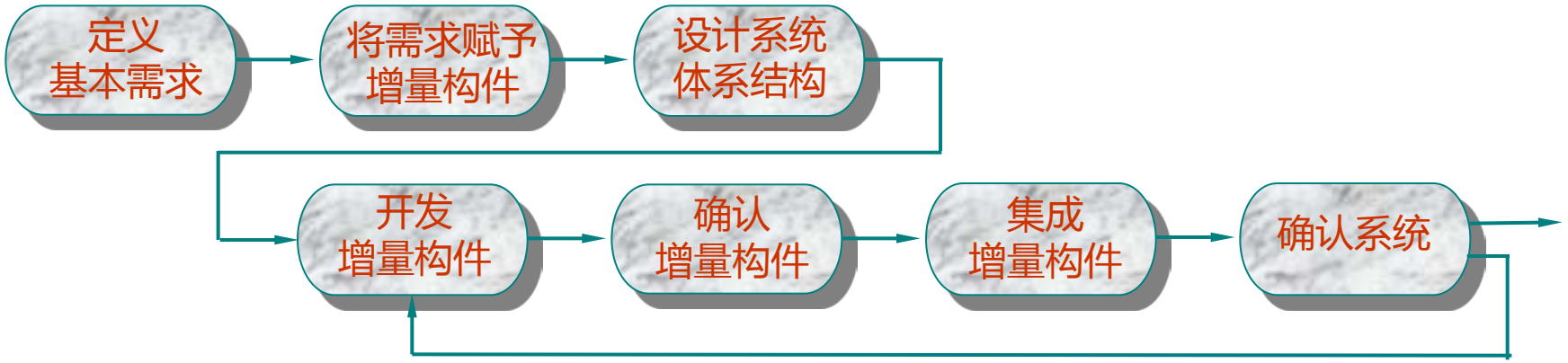
软件工程：螺旋模型

- 螺旋模型将瀑布模型与演化模型结合起来，并且加入两种模型均忽略了的风险分析。
- 螺旋模型沿着螺线旋转，自内向外每旋转一圈便开发出更完善的一个新版本。
  - 制定计划：确定软件目标，选定实施方案，弄清项目开发的限制条件；
  - 风险分析：分析所选方案，考虑如何识别和消除风险；
  - 实施工程：实施软件开发
  - 客户评估：评价开发，提出修正建议。



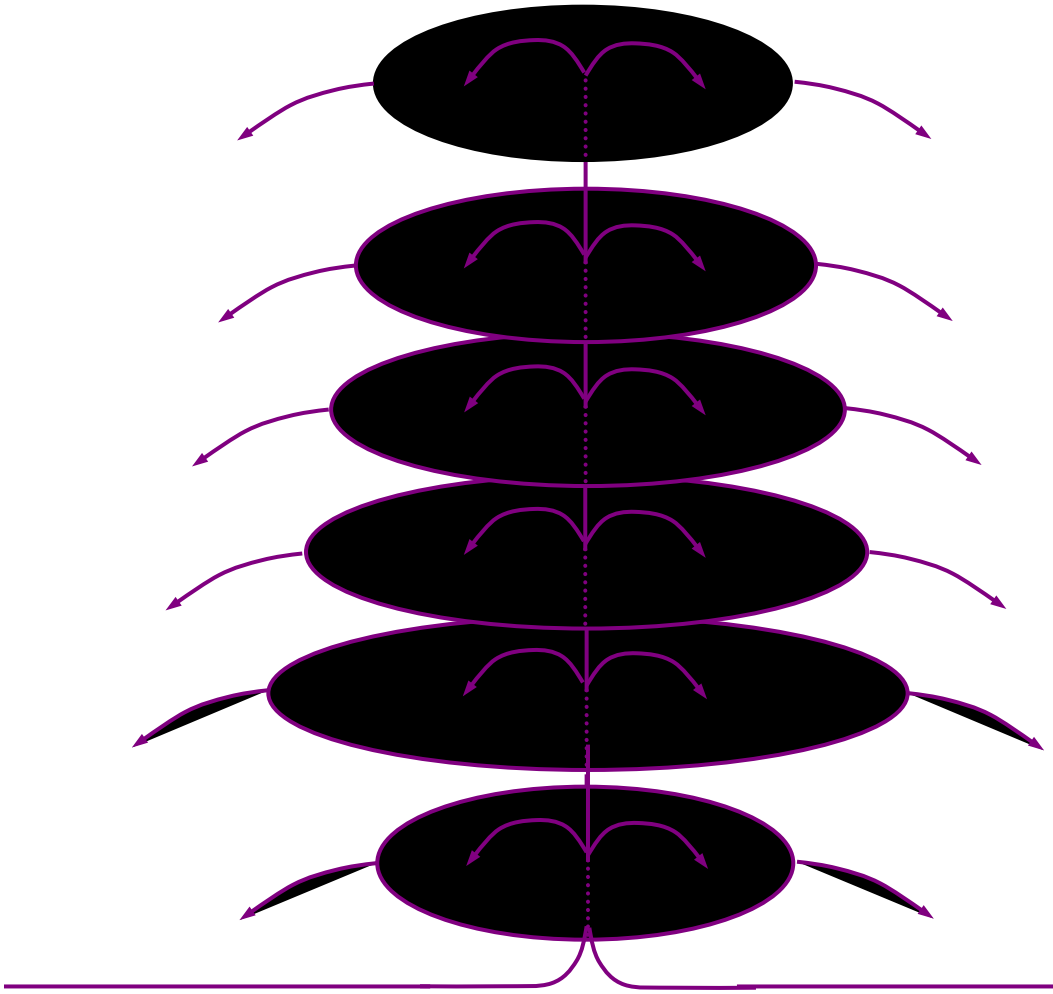
软件工程：增量模型

- 把软件产品分解成一系列的增量构件，在增量开发迭代中逐步加入。
- 每个构件由多个相互作用的模块构成，并且能够完成特定的功能。
- 增量开发方法的新演进版本叫做“极限程序设计（eXtreme Programming）”。



软件工程：喷泉模型

- 体现了迭代和无间隙的特性。
- 系统某个部分常常重复工作多次，相关对象在每次迭代中随之加入演进的软件成分。
- 无间隙是指在各项开发活动，即分析、设计和编码之间不存在明显的边界。
- 喷泉模型是对象驱动的过程。



维护与演进阶段

集成与测试阶段

编程阶段

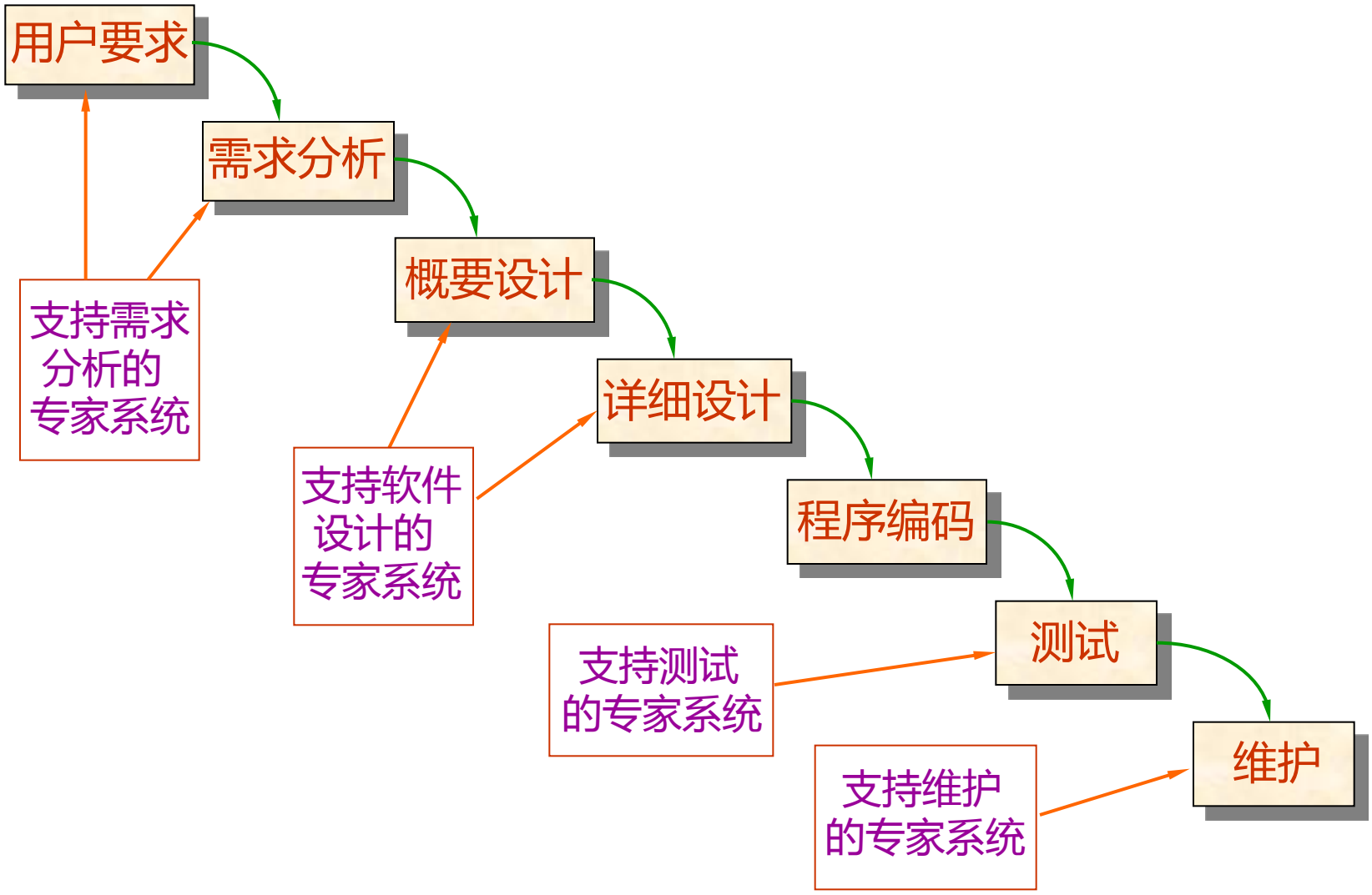
设计阶段

分析阶段

需求阶段

软件工程：智能模型

- 智能模型是基于知识的软件开发模型，它把瀑布模型和专家系统综合在一起。
- 该模型在各个开发阶段都利用了相应的专家系统来帮助软件人员完成开发工作。
- 为此，建立了各个阶段的知识库，将模型、相应领域知识和软件工程知识分别存入数据库。以软件工程知识为基础的生成规则构成的专家系统与包含应用领域知识规则的其他专家系统相结合，构成该应用领域的开发系统。


用户要求
需求分析
概要设计
详细设计
程序编码
测试
维护
支持需求分析的专家系统
支持软件设计的专家系统
支持测试的专家系统
支持维护的专家系统

软件工程

- 在软件开发过程中必须遵循的软件工程原则有：
  - 抽象与自顶向下、逐层细化
  - 信息隐蔽和数据封装
  - 模块化
  - 局部化
  - 确定性
  - 一致性和标准化
  - 完备性和可验证性

软件工程

- 软件工程的基本原理有：
  - 按软件生存期分阶段制定计划并认真实施；
  - 坚持进行阶段评审；
  - 坚持严格的产品控制；
  - 使用现代程序设计技术；
  - 明确责任，使得工作结果能够得到清楚的审查；
  - 用人少而精；
  - 不断改进开发过程。

匈牙利命名法

- 由Microsoft匈牙利程序员查尔斯·西蒙尼(Charles Simonyi) 提出：
- 变量名＝属性＋类型＋对象描述
  - 每一对象的名称都要求有明确含义，可以取对象名字全称或名字的一部分。
  - 命名要基于容易记忆容易理解的原则。
  - 保证名字的连贯性是非常重要的。

命名规则

- 变量名＝属性＋类型＋对象描述
  - (1)属性部分：
    - 全局变量： g，常量 ： c
  - (2)类型部分：
    - 句柄： h，布尔型： b，浮点型： f ，无符号： u，图像：im
  - (3)描述部分：
    - 初始化： Init，临时变量： Tmp，目的对象： Dst ，源对象： Src，窗口： Wnd
  - 例如：gimCoverImage

命名规则

- 变量的名字应该能够反应他们的意义或者用途。
- 变量名应该以大写字母开头的大小写混合形式
  - 例如：**imLena, QualityofLife**
- 不使用下划线，**MATLAB** 的**Tex** 解释程序会将其翻译为下标转换符。

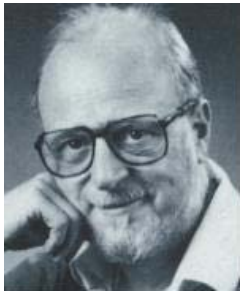
命名规则

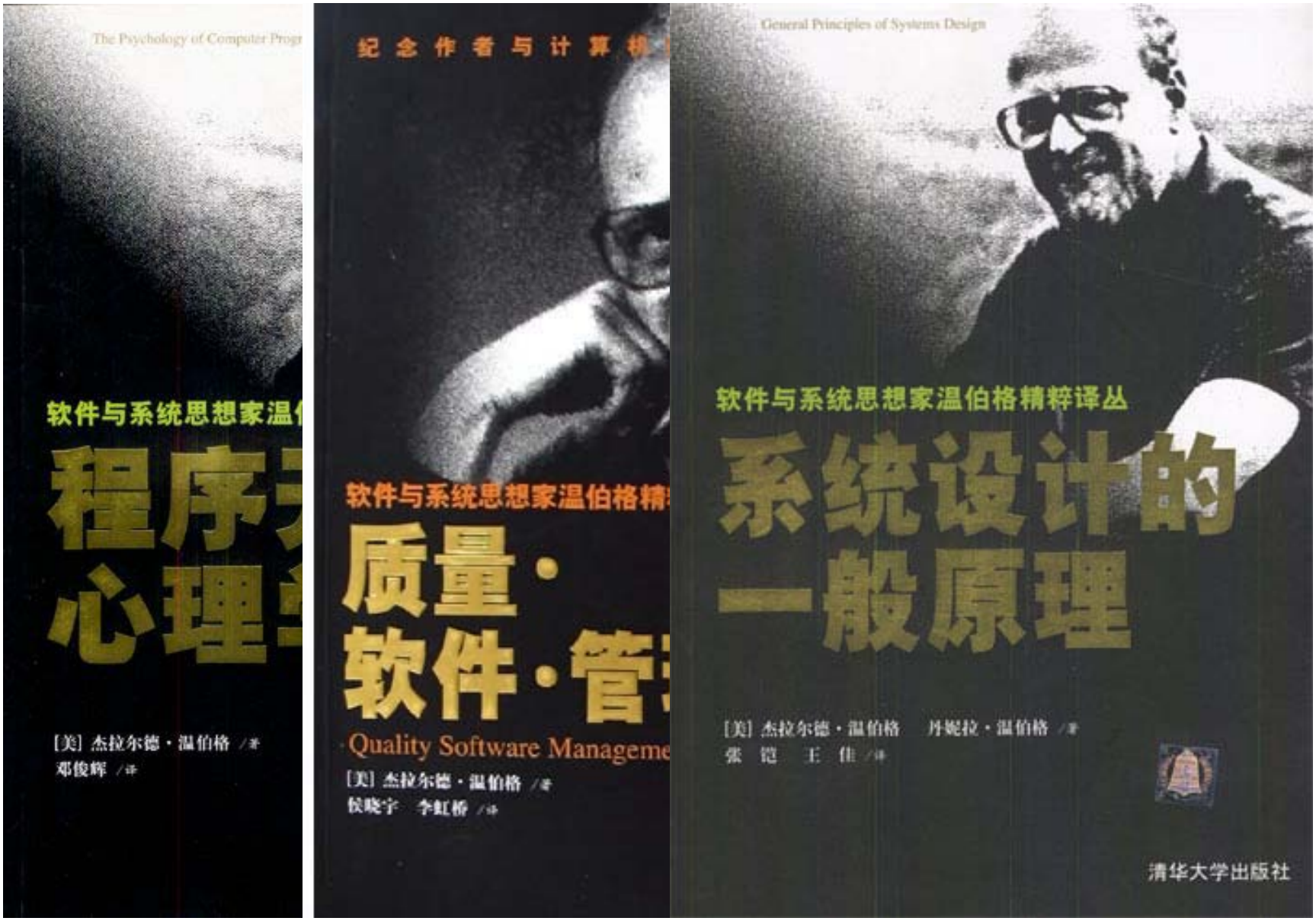
- 应用范围比较大的变量应该具有有意义的变量名，小范围应用的变量应该用短的变量名；
- 作为“草稿变量”的临时存储空间或者索引可以用短名字。程序员在读到这样的变量的时候，可以假定这个变量的值在没有几行之后的代码中就不会再用到。
- 循环变量应该以i、j、k 等为前缀

```
for iFile = 1：nFiles
…
end
```

温伯格 (Gerald M. Weinberg)

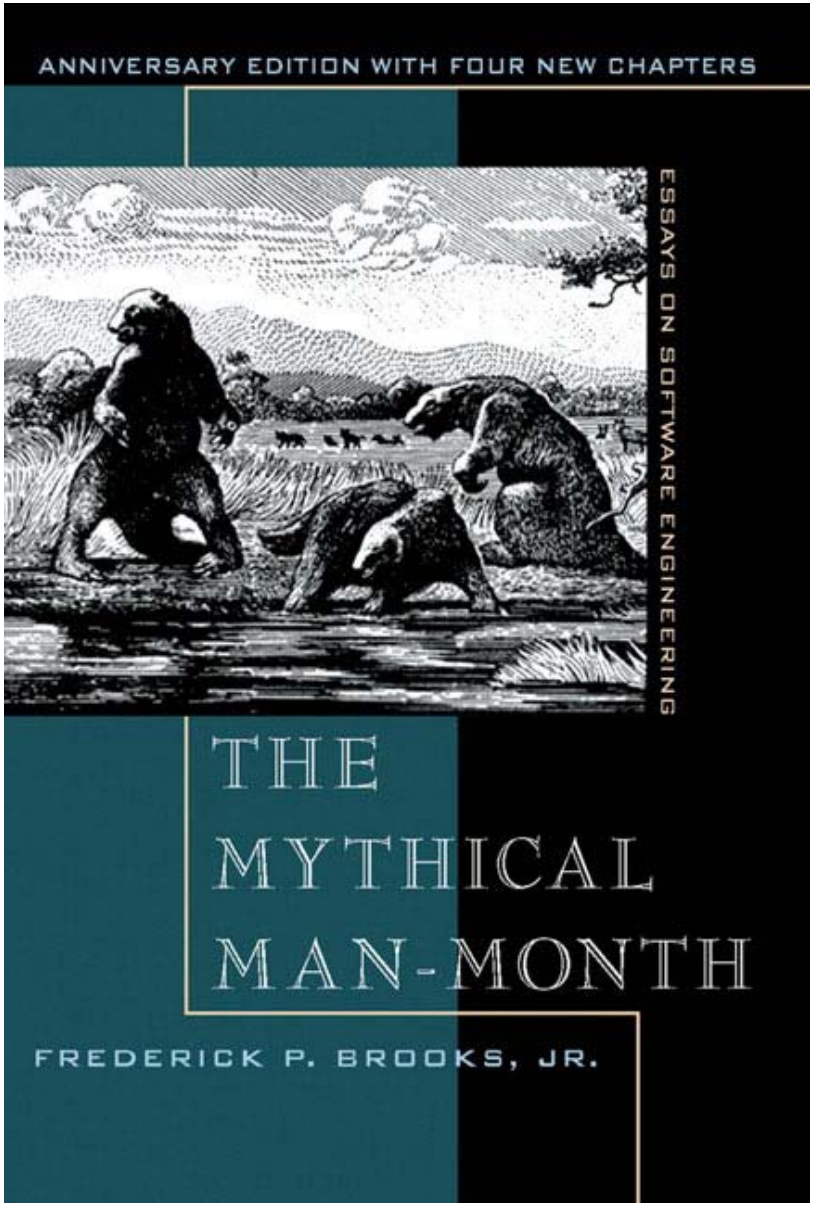
- 温伯格首要的贡献集中于软件领域，他是从个体心理、组织行为和企业文化角度研究软件管理和软件工程的权威和代表人物。在超过40年的软件职业生涯中，温伯格从事过软件开发，软件项目管理、软件管理教学和咨询，他更是一位杰出的软件专业作家和思想家。
- 1997年，温伯格因其在软件领域的杰出贡献，被美国计算机博物馆的计算机名人堂选为首批5位成员之一。

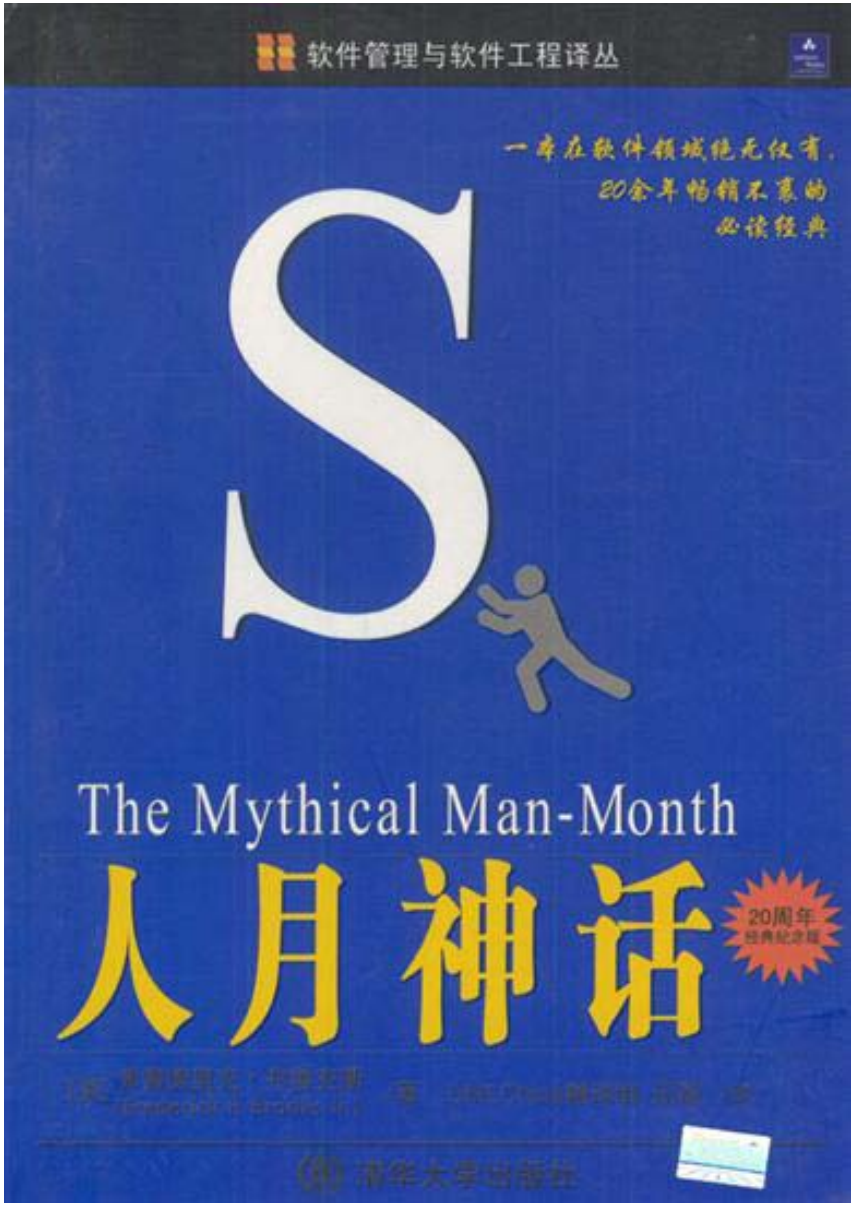




布鲁克斯 (Frederick P Brooks)

- Frederick P．Brooks，Jr．曾荣获美国计算机领域最具声望的图灵奖桂冠。美国计算机协会(ACM)称赞他“对计算机体系结构、操作系统和软件工程作出了里程碑式的贡献”。
- Brooks博士是北卡罗莱纳大学KENAN-FLAGLER商学院的计算机科学教授。
- 他被认为是“IBM 360系统之父”，曾担任360系统的项目经理，以及360系统项目设计阶段的经理。凭借在此项目中的杰出贡献，他与Bob Evans和Erich Bloch在1985年荣获了美国国家技术奖(Nationaliedal of Technology)。Brooks博士早期曾担任IBM公司Stretch和Harvest计算机的体系结构设计师。





布鲁克斯 (Frederick P Brooks)

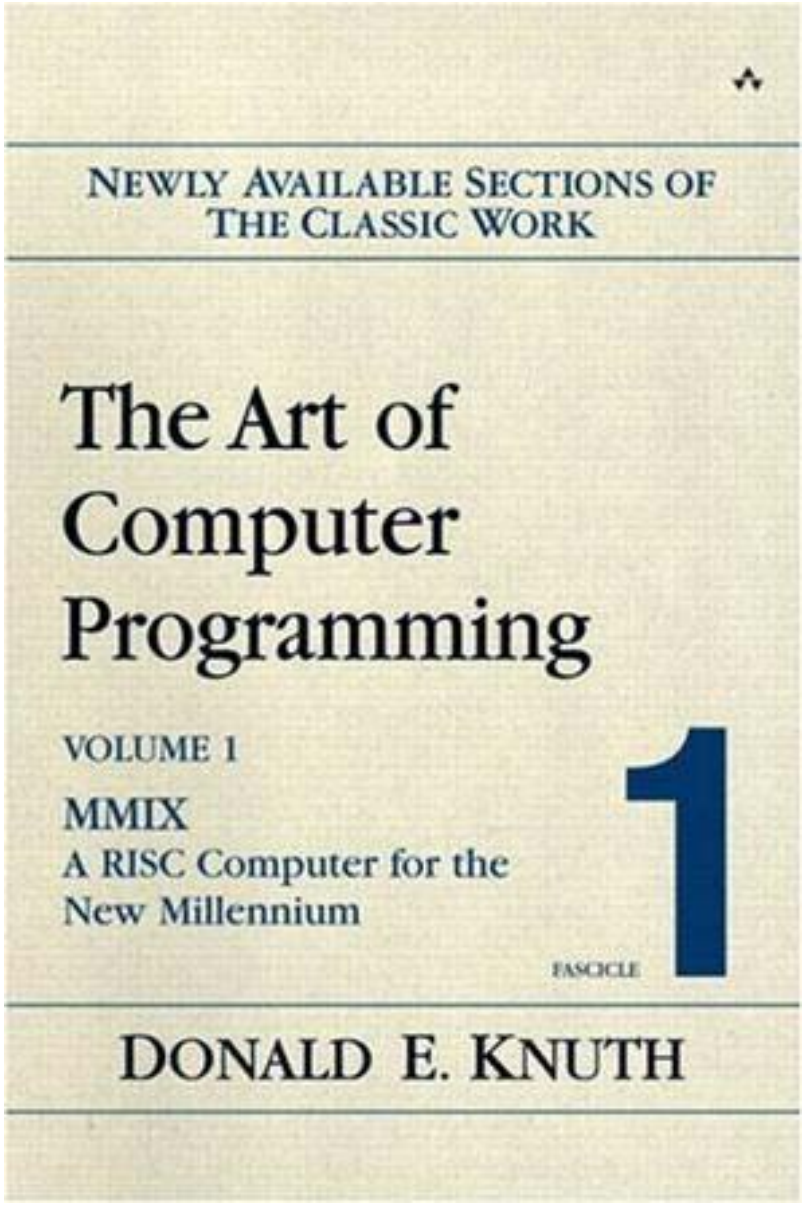
- 在延期的课题中加入人力只能让它延迟更多。
- 怀一个孩子需要九个月，无论你把任务分给多少个女人。

克努斯(Donald Knuth)

- Donald.E.Knuth：算法和程序设计技术的先驱者，是计算机排版系统TEX和METAFONT的发明者。
- 斯坦福大学计算机程序设计艺术的荣誉退休教授，Knuth教授获得了许多奖项和荣誉，包括美国计算机协会图灵奖，美国前总统卡特授予的科学金奖(MedalcofcScience)，美国数学学会斯蒂尔奖(AMScSteelecPrize)，以及1996年11月由于发明先进技术荣获的极受尊重的京都奖(KyotoPrize)．他因这些成就和大量创造性的影响深远的著作(19部书和160篇论文)而誉满全球。

克努斯(Donald Knuth)

- 用量化的方式思考问题
- 我同一时间内只做一件事情，这就是计算机科学家所谓的批处理——另一种方式是不停切换，但我不那样做。



四大数学软件

- 1. Matlab
- 2. Mathematica
- 3. Maple
- 4. MathCAD

Matlab Programming Tips and Tricks

Samuel Cheng

University of Oklahoma-Tulsa

Myth

- Matlab is slow
  - Think again!
  - Matlab is extremely fast for matrix computation
    - Matlab is an encapsulation of highly optimized Lapack and Blas numerical library

Experiment 1

- **Matrix multiplication for matrices of size 500 × 500**
- **C: 4.08 sec**

```
size=500;
for (i=0;i<size;i++)
 for (j=0;j<size;j++)
  for (k=0;k<size;k++)
            c[i][j]=a[i][k]*b[k][j];
```

- **Matlab: 0.53 sec**

Don’t expect your tiny C code will beat Matlab for matrix operations! Matlab is slow only when you make it so!

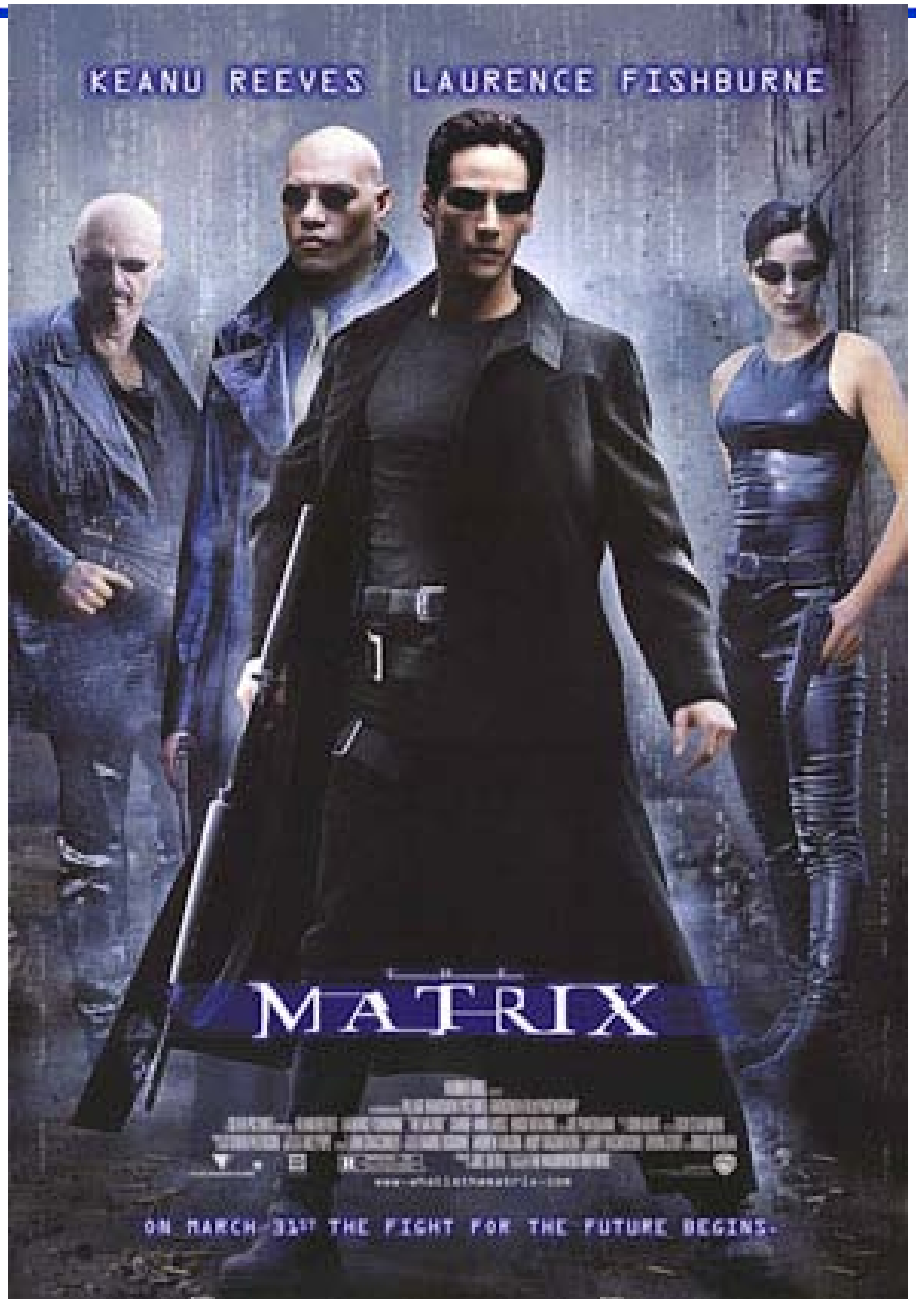
Flow Control

- Matlab is a fully power programming language with flow control capability of other high level languages
  - If-then-else
  - for-end
  - while-end
  - try-catch-end

In most cases, you don't need them at all if you do things right!

Matlab

- Matlab is not “Math” lab
- It is Matrix lab
- Think everything in matrix


KEANU REEVES
LAURENCE FISHBURNE
MATRIX
ON MARCH 31ST THE FIGHT FOR THE FUTURE BEGINS.

Vectors and Matrices

- **Row vectors**

```
>> X=[1,3,5,7,9];
```

- **Column vectors and transpose**

```
>> X=[1;3;5;7;9];
>> X=[1,3,5,7,9]';
```

$$X = \begin{bmatrix} 1 \\ 3 \\ 5 \\ 7 \\ 9 \end{bmatrix}$$

- **Matrices**

```
>> X=zeros(2); X=[0,0;0,0];
>> X=ones(2); X=[1,1;1,1];
>> X=eye(2); X=[1,0;0,1];
```

$$X = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Colon

```
>> X=1:5; % X=[1 2 3 4 5]
```


```
>> X=0:0.3:1; % X=[0 0.3 0.6 0.9]
```

```
>> X=ones(2)
>> X(:)
ans =
        1
        1
        1
        1
```

Dot Products

- Every operation in Matlab is matrix operation by default
- Non-matrix operation is generalized to matrix operation as element-wise operation (`sin`, `cos`, etc.)
- To convert matrix operation to element-wise operation, add dot in front (`.*`,`.^`)

```
>> A=ones(2); A^2
ans =
     2     2
     2     2
```

```
>> A=ones(2); A.^2
ans =
     1     1
     1     1
```

Most Important Tip

- Avoid loops by all means
- E.g. y=sin(x), x=0.1,0.2,...,2*pi
- Stupid code:

```
for i=1:2*pi*10          x=0.1*i;
y(i)=sin(x);  end
```

- Better:

```
x=0.1:0.1:2*pi;          y=sin(x);
```

ndgrid

- **Bad**

```
>> xid=1;yid=1;
>> for x=-10:0.1:10
for y=-10:0.1:10
z(xid,yid)=x^3+y^2;
yid=yid+1;
end
xid=xid+1;
end
```

- **Much better**

```
>> [x,y]=ndgrid(-10:0.1:10,…
-10:0.1:10);
z=x.^3+y.^2;
```

Batch File

- Batch file in Matlab is simple. Simply put your code in a file with extension .m
- Every symbols will be visible in the current workspace
- Good for initial development

function

- **Functions are exactly the same as batch files except they hide all variables except those you return**

- Don’t assume dimensions of your inputs (don’t assume your input to be a column or row vector)
- Break your function into meaningful functions (a well-design program needs very few comments)
- Use functions whenever possible (instead of batch)

function

- **Exercise: can you write the function in 1 line?**

```
>> type isprime1.m
function bool=isprime1(n)
bool=1;
for i=2:floor(sqrt(n))
        if mod(n,i)==0
                bool=0; break;
        end
end
>> bool=isprime1(n);
>> i
??? Undefined function or variable ‘i’.
```

Answer

```
>> type isprime2.m
function bool=isprime2(n)
bool=all(mod(n*ones(floor(sqrt(n))-1,1)',…
 2:floor(sqrt(n))));
```

function pointer

```
>> add=@(x,y) x+y
add =
    @(x,y)x+y

>> add(1,2)
ans =
    3

>> addone=@(x) add(1,x);
>> addone(3)
ans =
    4
```

size / length

```
>> X=ones(4,5);
>> size(X)
ans =
        4        5
>> size(X,2)
ans =
        5
>> size(X(:))
ans =
        20       1
>> length(X(:))
ans =
        20
```

size / length

```
>> type horrible.m
X=imread('mypic.png');
nX=320;
nY=200;
process(X,nX,nY);
…
```

```
>> type better.m
X=imread('mypic.png');
nX=size(X,1);
nY=size(X,2);
process(X,nX,nY);
…
```

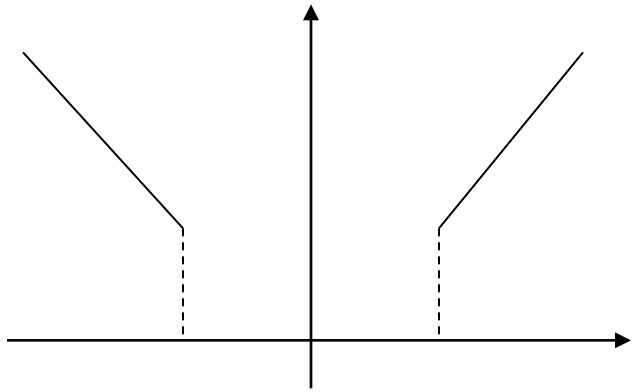
```
>> type best.m
X=imread('mypic.png');
process_new(X);
…
```

Don't put fixed values all over your codes! "I'll never change their values" is not an excuse

find

- **E.g. thresholding**
- **Bad**

```
function x=threshold(x,th)
for xid=1:length(x)
        if abs(x(xid))<th
                x(xid)=0;
        end
```



$$y = \begin{cases} x & \text{if } |x| \geq th \\ 0 & \text{otherwise} \end{cases}$$

- **Better**

```
function x=threshold(x,th)
ind=find(abs(x)<th);
x(ind)=0;
```

- 1-line

```
x(abs(x)<th)=0
```

exist

```
>> type threshold.m
function x=threshold(x,th)
if ~exist('th','var')
        th=1;
end
x(abs(x)<th)=0;
```

Use exist to assign default variables for your functions

sprintf and eval

```
>> type bad.m
Process(b1);
Process(b2);
Process(b3);
Process(b4);
Process(b5);
```

General rule:

Write codes with low “entropy” (deterministic)

```
>> type good.m
for id=1:5
 eval(sprintf(‘Process(b%d);’,id));
end
```

cell

- **Cell arrays / matrices can store mixed content for each component**

```
>> a={1,1:2,'a'}
a =
        [1]      [1x2 double]  'a'
>> a(2)
ans =
        [1x2 double]
>> a{2}
ans =
        1        2
```

File I/O

- **fopen**
- **fread**
- **fwrite**
- **fprintf**

More or less same as C

- **save/load**

```
>> save tmp a b c
>> load tmp
>> save tmp2 a -ascii
```

Debugging

- Use dbstop to initiate debugger
- Use dbclear all to erase all stop points
- Shortcuts
  - F10 – step over
  - F11 – step in
  - Shift-F11 – run to next stop point
  - F12 – toggle stop point

try/catch

```
>> x=rand(1,99);
>> for i=1:100;
y(i)=x(i)^2;
end
??? Index exceeds matrix dimensions.
```

```
>> for i=1:100;
        try
                y(i)=x(i)^2;
        catch
                i
        end
end

i =
        100
```

Profiling

- **Using `tic/toc`**

```
>> A=rand(500);
>> tic; A*A; toc;
Elapsed time is 0.329000 seconds.
```

- **Matlab has its own profiling tools**
  - **`help profile`**

Interface with C

- Matlab can be slow with non-matrix operation
- Look for internal function in those cases
- Speed up by interfacing with C

Interface with C

- Write out file
- Call C using dos
- Read file

```
>> type interface.m
function Imout=interface(size,parameter)
Im=generate_img(size);
imwrite(Im,'tmp.png');
dos(sprintf('process tmp.png %d',parameter));
Imout=imread('tmp.png');
```

Using Mex

```
>> type mex_skeleton.c
#include "mex.h"

void mexFunction(int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])
{
 double *y,*x1,*x2;
 int nrows,ncols;
 nrows = mxGetM(prhs[0]);
 ncols = mxGetN(prhs[0]);

 plhs[0] = mxCreateDoubleMatrix(nrows,ncols,mxREAL);

 x1 = mxGetPr(prhs[0]);
 x2 = mxGetPr(prhs[1]);
 y = mxGetPr(plhs[0]);

 process(x1,x2,y);
}
```

Using Mex

```
>> mex mex_skeleton.c
>> x1=rand(1,10);
>> x2=rand(1,10);
>> y = mex_skeleton(x1,x2);
```

Visualization Functions

- ezplot
- plot
- plot3
- mesh
- surf
- …
- See help document

Most Common FAQs for Figures

- Don't show axis

  >>axis off

- Change axis ranges

  >> axis([xstart xend ystart yend]);

- Change axis font size

  Click axis

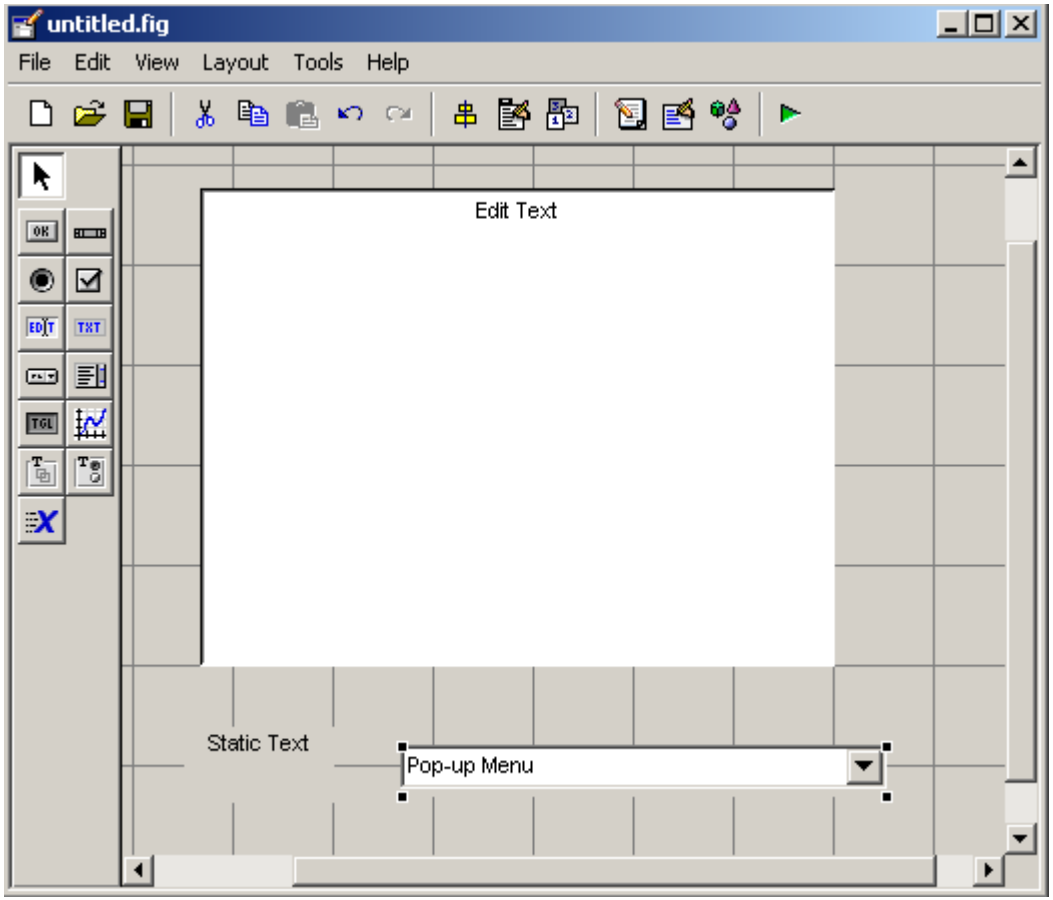
  >> set(gca,'fontsize',fontsize);

- Use Latex

  >> title('$\frac{x}{y}','interpreter','latex');

GUI

- **Help guide**



Conclusions

- Matlab can be fast in terms of both development and actual running, an extremely good tool for prototyping
- Avoid loop in all means
- Getting help
  - Use “help” often
  - Google “matlab command”
  - Email me or knock my door (try afternoon and evening)

FAQs

- 关于运行速度：避免反复分配内存空间

```
N = 5e3;
x = zeros(N,1);
x(1) = 1000;
for n=2:N
    x(k) = 1.05*x(k-1);
end
```

FAQs

- 关于运行速度：以下两段程序有何差异（test1.m）？

```
N = 5e3;
x = randn(N,N); y = zeros(N,N);
tic;
for r = 1:N % for each fixed row
    for c = 1:N % column-wise
      y(r,c) = x(r,c);
    end
end
toc

tic;
for c = 1:N % for each fixed column
    for r = 1:N % row-wise
      y(r,c) = x(r,c);
    end
end
Toc
```

FAQs

- 如何将Matlab生成的图贴到Word文件中

  Edit→copy options：线条图用Matafile；图像用Bitmap

- 如何使图像显示时不出现多余的边框？

```
>> iptsetpref('ImshowBorder','tight');
```

FAQs

- **Matlab**中下标、斜体及希腊字母的使用方法
  - 下面给出**Matlab**中下标及希腊字母的使用方法，还有更多的使用方法可以参考**matlab**帮助文档中的**Text Properties:**

- 下标用 _(下划线)
- 上标用^ （尖号）
- 斜体 \it
- 黑体 \bf
- 希腊字母等特殊字符用 \加拼音如
- α \alpha
- β \beta
- γ \gamma
- θ \theta
- Θ \Theta
- Γ \Gamma
- δ \delta
- Δ \Delta
- ξ \xi
- Ξ \Xi
- η \elta
- ε \epsilong
- ζ \zeta
- μ \miu
- υ \nu
- τ \tau
- λ \lamda
- Λ \Lamda
- π \pi
- Π \Pi
- σ \sigma
- Σ \Sigma
- φ \phi
- Φ \Phi
- ψ \psi
- Ψ \Psi
- χ \chi
- ω \ommiga
- Ω \Ommiga
- < \leq
- > \geq
- 不等于 \neq
- << \ll
- >> \gg
- 正负 \pm
- 左箭头 \leftarrow
- 右箭头 \rightarrow
- 上箭头 \uparrow
- 上圆圈（度数） \circ
- 例 text(20,30,'\alpha_2^\beta')
- 注： 可用{ }把须放在一起的括起来

FAQs

- Matlab中使用Plot函数动态画图方法总结

- 一. AXIS 移动坐标系
  - 这种方法是最简单的一种方法，适合于数据已经全部生成的场合，先画图，然后移动坐标轴。

```
%先画好，然后更改坐标系
%在命令行中 使用 Ctrl+C 结束
t=0:0.1:100*pi;
m=sin(t);
plot(t,m);
x=-2*pi;
axis([x,x+4*pi,-2,2]);
grid on
while 1
   if x>max(t)
      break;
   end
   x=x+0.1;
   axis([x,x+4*pi,-2,2]); %移动坐标系
   pause(0.1);
end
```

- 二. Hold On 模式
  - 此种方法比较原始，适合于即时数据，原理是先画上一帧，接着保留原始图像，追加下一幅图像，此种方式比较繁琐，涉及画图细节，并且没有完整并连续的Line对象数据。

```
%%
% Hold On 法
% 此种方法只能点，或者分段划线
hold off
t=0;
m=0;
t1=[0 0.1]; %要构成序列
m1=[sin(t1);cos(t1)];
p = plot(t,m,'*',t1,m1(1,:),'-r',t1,m1(2,:),'-b','MarkerSize',5);
x=-1.5*pi;
axis([x x+2*pi -1.5 1.5]);
grid on;

for i=1:100
   hold on
   t=0.1*i;  %下一个点
   m=t-floor(t);
   t1=t1+0.1; %下一段线(组)
   m1=[sin(t1);cos(t1)];
   p = plot(t,m,'*',t1,m1(1,:),'-r',t1,m1(2,:),'-b','MarkerSize',5);
   x=x+0.1;
   axis([x x+2*pi -1.5 1.5]);
   pause(0.01);
end
```

➢ 三. Plot 背景擦除模式

- ◆这种模式比较适合画动画，效率比较高，刷新闪烁小，适合即时数据，最终的Line结构数据完整。
- ◆了解此方法之前要搞清楚 Plot函数的原型是什么：Plot函数，输入为 X-Y (-X)坐标元组、以及“属性”-“值对，输出为一个列向量（每条曲线岁对应的Line结构 Handle，每一行代表一个 线条的handles），每一线条都有 XData，YData 向量。如果你画了2条线，那么会返回 2×1的向量。
- ◆重新画图不需要 重新书写 Plot，只需要 刷新图像即可，使用drawnow函数。

➢ 1. 画一个点的动画：

```
%%
%采用背景擦除的方法，动态的划点，并且动态改变坐标系
% t,m 均为一行 ，并且不能为多行
t=0;
m=0;
p = plot(t,m,'*',...
  'EraseMode','background','MarkerSize',5);
x=-1.5*pi;
axis([x x+2*pi -1.5 1.5]);
grid on;

for i=1:1000
   t=0.1*i;       %两个变量均不追加
   m=sin(0.1*i);
   set(p,'XData',t,'YData',m)
   x=x+0.1;
   drawnow
   axis([x x+2*pi -1.5 1.5]);
   pause(0.1);
end
```

- 2. 动态多条曲线(即时数据)
-

```
%采用背景擦除的方法，动态的划线，并且动态改变坐标系多行划线

t=[0]
m=[sin(t);cos(t)]
p = plot(t,m,...
  'EraseMode','background','MarkerSize',5);
x=-1.5*pi;
axis([x x+2*pi -1.5 1.5]);
grid on;

for i=1:1000
   t=[t 0.1*i];                %Matrix 1*(i+1)
   m=[m [sin(0.1*i);cos(0.1*i)]]; %Matrix 2*(i+1)
   set(p(1),'XData',t,'YData',m(1,:))
   set(p(2),'XData',t,'YData',m(2,:))
   drawnow
   x=x+0.1;
   axis([x x+2*pi -1.5 1.5]);
   pause(0.5);
end
```

在MATLAB7.0中编译和发布可执行文档

- 1.安装C编译器（前提是您的电脑已安装了VC）

```
>>mbuild -setup
```

```
>> mbuild -setup
Please choose your compiler for building standalone MATLAB applications:

Would you like mbuild to locate installed compilers [y]/n? y

Select a compiler:
[1] Lcc C version 2.4.1 in C:\PROGRAM FILES\MATLAB71\sys\lcc
[2] Microsoft Visual C/C++ version 6.0 in C:\Program Files\Microsoft Visual Studio

[0] None

Compiler: 2

Please verify your choices:

Compiler: Microsoft Visual C/C++ 6.0
Location: C:\Program Files\Microsoft Visual Studio

Are these correct?([y]/n): y

Try to update options file: C:\Documents and Settings\XPMUser\Application Data\MathWorks\MATLAB\R14SP3\compopts.
From template:              C:\PROGRAM FILES\MATLAB71\BIN\win32\mbuildopts\msvc60compp.bat

Done . . .


--> ""C:\Program Files\MATLAB71\bin\win32\mwregsvr" "C:\Program Files\MATLAB71\bin\win32\mwcomutil.dll""

DllRegisterServer in C:\Program Files\MATLAB71\bin\win32\mwcomutil.dll succeeded

--> ""C:\Program Files\MATLAB71\bin\win32\mwregsvr" "C:\Program Files\MATLAB71\bin\win32\mwcommgr.dll""

DllRegisterServer in C:\Program Files\MATLAB71\bin\win32\mwcommgr.dll succeeded
```

- **2.在MATLAB中，将M文档编译成exe（可执行）文档**

  >>mcc -m guimcc

  - **To get started, select MATLAB Help or Demos from the Help menu.**

  - **Warning: No matching builtin function available for D:\MATLAB7\toolbox\simulink\simulink\set_param.bi**

  - 上面警告的解决方法：
    - 将 **\MATLAB7\toolbox\compiler\deploy\matlabrc.m** 中的
    - **81行 set_param(0,'PaperType',defaultpaper);**
    - **82行 set_param(0,'PaperUnits',defaultunits);**
    - 注释掉之后，又重新编译了一次，没有警告信息出来了

- **3.将编译生成的文档发布到没有MATLAB的电脑上**
  - ◆在 **$MATLAB\toolbox\compiler\deploy\win32** 下找**MCRInstaller.exe**（**76M**左右）文档
  - ◆然后在没有装**MATLAB**的机子上安装**MCRInstaller**到 **D:\MATLAB Component Runtime(**目录中，最好不要有空格，如用 **D:\MCR** 就能够了**)**
  - ◆配置环境变量，右键点击“我的电脑”-》属性-》高级-》环境变量-》新建
    - ●变量名：**Path**
    - ●变量值：**D:\MCR\v70\runtime\win32**

- 然后运行exe文档，每次都是出现DOS界面后，几秒钟就自动关闭了。GUI界面一次都运行不出来，也没有错误信息提示。
- 解决办法：
- {因为MATLAB需要用到处理器的数学运算部分(MATLAB默认用的是INTEL的数学处理单元),故需要配置一下BLAS(Basic Linear Algebra Subroutines,就是"基础线性几何子程式"的意思)环境变量．首先请确认您的MATLAB的文档夹中有如下文档:atlas_Athlon.dll(AMD系列的请用这个)， atlas_P4.dll(P4的用这个)，atlas_PIII.dll(P3的用这个)，atlas_PII.dll(P2的用这个)，这些是对应处理器的数值运算优化文档 }
- 先找到 $MATLAB7\bin\win32目录下的atlas_Athlon.dll（AMD系列CPU使用）和atlas_P4.dll（奔4用）文档，放到MCR目录中（随便），然后配置环境变量
  - 变量名：BLAS_VERSION
  - 变量值：D:\MCR\v70\atlas_Athlon.dll
- 这样配置以后，再运行exe文档就OK了。

- 注意事项:
  - 编译过程中，很可能会有如下报错

```
>> mcc -m interface
To get started, select MATLAB Help or Demos from the Help menu.
??? Unable to locate close;contentwindow as a function on the MATLAB path
Warning: An object instance still exists.
Use the objectdirectory command to see a count of existing instances.
??? Depfun error: 'Unable to locate close;contentwindow as a function on the MATLAB path‘
```

  - 这种情况是因为将“close;contentwindow”语句写在按钮控件属性的“callback”里面了{但是假如只是单个语句的话，是可行的（比如：只有contentwindow）}。此时，只要将“close;contentwindow”语句写在M文档中按钮的“callback”里，再编译就不会出错了

总结

- 技巧不是来自一、两个人的深邃的洞察力，而是由许多细微的知识、信念、技能和初识者必须积累养成的习惯汇聚而成的。

真印良品
TrueMark.cn

Thanks

士不可以不弘毅，任重而道远