化工过程模拟及软件应用

杨鑫
化学化工学院
重庆理工大学
第一实验楼A203
cheyangxin@cqut.edu.cn

化工过程数值计算

- 简介
- 数据处理：插值、拟合
- 数值积分
- 线性方程组的求解
- 非线性方程（组）的求解
- 常微分方程（组）求解
- 最优化

非线性方程（组）求解

非线性方程

- 线性方程：未知数都是一次的方程。
- 线性方程一般形式：ax+by+…+cz+d=0
- 非线性方程：因变量与自变量之间的关系不是线性的方程。
- 非线性方程一般形式：f(x)= 0
- 方程的解称为方程的根或函数的零点，求解非线性方程问题比较复杂非线性方程，很多无法用解析的方法求出摩擦系数，只能用数值求解的方法
  - 求解高次方程组$7x^6-x^3+x-1.5 = 0$的根
  - 求解含有指数和正弦函数的超越方程$e^x-\sin(x) = 0$的零点。

化工中的非线性方程问题

- 求解非线性方程是化工设计及模拟计算中必须解决的一个问题
- 管路设计：已知雷诺数Re，求出摩擦系数λ。常见的雷诺数和摩擦系数关系方程在雷诺数低于4000时关系式

$$\left(\frac{1}{\lambda}\right)^{0.5}=1.74-2\lg\left[\frac{2\varepsilon_i}{d_i}+\frac{18.7}{Re\lambda^{0.5}}\right]$$

- 在n个组分的等温闪蒸计算中，通过物料和相平衡计算，得到方程（α未知，ki为相平衡常数，zi为进料组分的摩尔浓度，均已知）：

$$\sum_{i=1}^{n}\frac{z_i(1-k_i)}{k_i+\alpha}=0$$

非线性方程求解

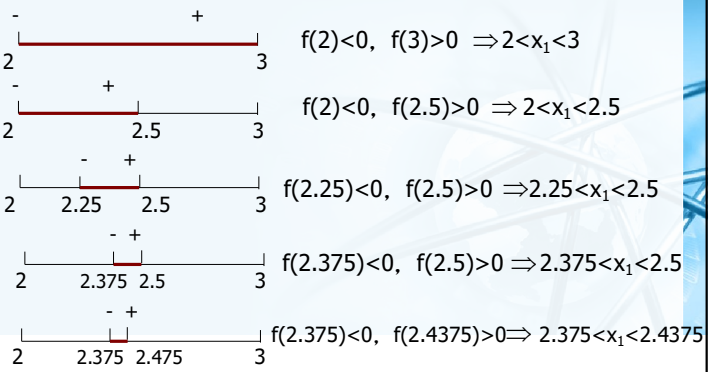
- 数值求解方法：
  - 二分法（bisection method）
  - 直接迭代法
  - 松弛迭代法
  - 牛顿迭代法…
- 非线性方程的根≥1，一种方法只能算出一个根。
- 在求解非线性方程时，要给定初始值或求解范围。
- 而对于具体的化工问题，初值和求解范围常常可根据具体的化工知识来决定。

求解方法：二分法

- 依据零点定理，确定方程的有根区间
  设 $f(x)\in C[a,b]$ ，且 $f(a)f(b)<0$ ，则方程 $f(x)=0$ 在区间 (a,b) 上至少有一个根。
- 不断二分有根区间，根据精确度得出近似解
- 简单、收敛速度慢

求解方法：二分法

求方程 $x^2-2x-1=0$ 的一个正的近似解（精确到**0.1**）



求解方法：直接迭代法

- 对给定的方程 f(x) =0，将它转换成等价形式：$x=\varphi(x)$
- 给定初值x_0，构造迭代序列

$$x_{k+1}=\varphi(x_k)$$

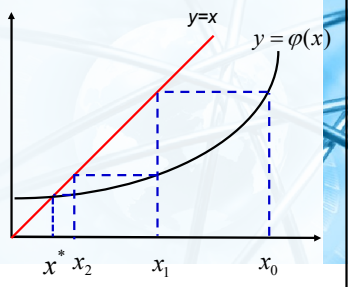
- 如果迭代收敛，

$$\lim_{k\to\infty}x_{k+1}=\lim_{k\to\infty}(x_k)=b$$

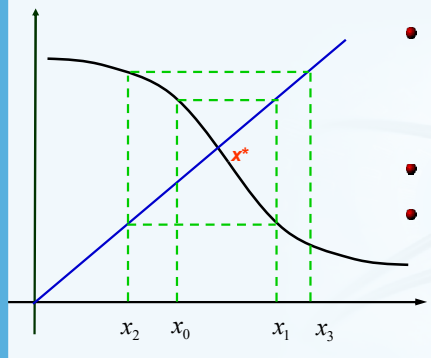
- $b=\varphi(b)$ 是方程f(x)=0的根
- 当 $|x_{k+1}-x_k|$ 小于给定的精度控制量时，$b=x_{k+1}$ 为方程的根

- 几何意义：交点的横坐标

$$x=\varphi(x)\Rightarrow\begin{cases}y=x\\y=\varphi(x)\end{cases}$$



直接迭代法



- 某些非线性方程用直接迭代法求解时，迭代过程是发散的。
- 解决办法： 松弛因子法
- 引入松弛因子 ω，通过选择合适的松弛因子ω，可使迭代过程收敛。

求解方法：松弛迭代法

- 松弛迭代法迭代公式 $x_{n+1}=x_n+\omega(\varphi(x_n)-x_n)$

- ω=1：松弛迭代变为直接迭代
- ω>1：迭代步长加大，加速迭代，但有可能使原来收敛的迭代变成发散
- 0<ω<1：迭代步长减小，适合于迭代发散或振荡收敛的情况，可使振荡收敛过程加速
- ω<0：使迭代反方向进行，可使一些发散迭代过程收敛

- 松弛法是否有效的关键：正确选定松弛因子ω的值
- ω的值根据经验选定。选用较小的松弛因子，一般可以保证迭代过程的收敛

求解方法： 牛顿（Newton）迭代法

- 借助于对非线性函数f(x)的泰勒展开（Taylor’s expansion），将非线性方程线性化
- 设x* 是方程$f(x)=0$ 的根，x_0为x* 附近的一个值
- 将$f(x)$ 在x_0附近做泰勒展开

$$f(x)=f(x_0)+(x-x_0)f'(x_0)+\frac{1}{2}(x-x_0)^2f''(\xi),\quad \xi\in[x,x_0]$$

- 令 $x=x^*$

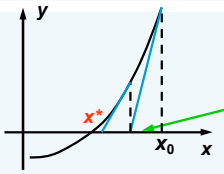
$$0=f(x^*)=f(x_0)+(x^*-x_0)f'(x_0)+\frac{1}{2}(x^*-x_0)^2f''(\xi)$$

- 取 $x_0\approx x^*$，将 $(x^*-x_0)^2$ 看成高阶小量：

$$0=f(x^*)\approx f(x_0)+f'(x_0)(x^*-x_0)$$

$$x^*\approx x_0-\frac{f(x_0)}{f'(x_0)}$$

牛顿迭代法的几何意义



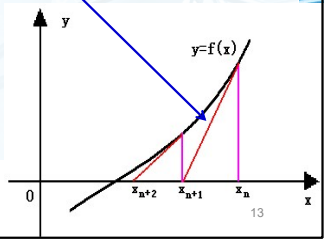
$$x^* \approx x_0 - \frac{f(x_0)}{f'(x_0)}$$

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$$

$$\lim_{k\to\infty} x_k = x^*$$

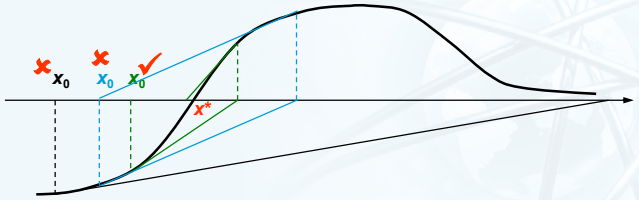
- 过 $(x_n, f(x_n))$ 的切线为 $y = f(x_n) + f'(x_n)(x - x_n)$
- 该切线与 $y = 0$ 的交点为 $x = x_{n+1}$

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$



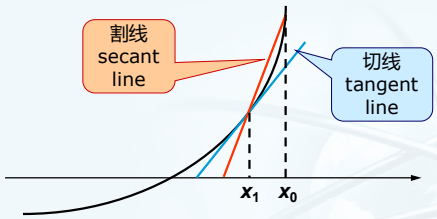
Newton迭代法的几何意义

- 牛顿法收敛性依赖于x_0 的选取
- 选取得当，收敛速度快
- 需计算函数的导数 f′ ，对复杂函数，不便于求导



正割法 (Secant Method)

- 对不便于求导的方程，用商差代替牛顿法里的函数导数



- 切线斜率 ≈ 割线斜率 ⇒ $f'(x_k) \approx \frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}}$

⇒ $x_{k+1} = x_k - \frac{f(x_k)(x_k - x_{k-1})}{f(x_k) - f(x_{k-1})}$ 需要2个初值 x_0 和 x_1

牛顿迭代法计算实例

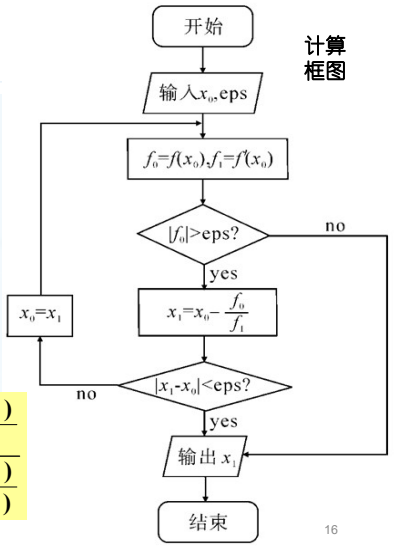
n个组分的等温闪蒸计算：

$$\sum_{i=1}^{n} \frac{z_i(1 - k_i)}{k_i + \alpha} = 0$$

- α是未知数
- k_i为相平衡常数，z_i为进料组分的摩尔浓度，均为已知数。
- 采用牛顿迭代公式，则可以得到如下的具体迭代公式：

$$a^{n+1} = a^n + \frac{\sum_{i=1}^{m} \frac{z_i(1 - k_i)}{k_i + a^n}}{\sum_{i=1}^{m} \frac{z_i(1 - k_i)}{(k_i + a^n)}}$$

计算框图



Mworks求解非线性方程：fzero

- fzero非线性函数的根函数库: TyOptimization
- 此算法由 T. Dekker 创建，它结合使用了对分法、正割法和逆二次插值方法
- x,fval,exitflag,output = fzero(fun,x0)
- 尝试求出 fun(x) = 0 的点 x、fun(x)（在 fval 输出中）、对 fzero 停止的原因编码的 exitflag，以及包含有关求解过程的信息的输出结构体
  - fun:单变量函数
  - **x0:初值，选择重要**
  - x：返回的方程的根
  - fval：算法收敛后的函数值
  - exitflag：对退出条件编码的整数（=1的话表明函数收敛于解x）
  - output — 有关优化过程的信息

求x^2-2x-1=0的根

```
f=x->x.^2-2*x-1
x=fzero(f,2)
```

结果：x=2.4142

Mworks求解多项式的根：roots

- 求解多项式的根：roots
- 以列向量的形式返回 p 表示的多项式的根
- 函数库: TyMath
- 多项式：$P(x)=p_1x^n+p_2x^{n-1}+\ldots+p_nx+p_{n+1}$
- r = roots(p)
- p: 多项式系数 $[p_1,p_2,\ldots p_{n+1}]$
- r: 多项式的根
- 注意：
- p只能写成向量$[p_1,p_2,\ldots p_{n+1}]$
- p不能写成矩阵$[p_1\ p_2\ \ldots\ p_{n+1}]$

求x^2-2x-1=0的根

```
>> p=[1,-2,-1]
>> r=roots(p)
r =
   -0.4142
    2.4142
```


例题

- 用维里方程计算200℃，1.013Mpa的异丙醇蒸汽的摩尔体积V与压缩因子Z。
- 异丙醇的维里系数为B=-388cm³/mol, C=-26000cm⁶/mol².
- 解：维里方程 $Z=1+\frac{B}{V}+\frac{C}{V^2}$ $\frac{pV}{RT}=1+\frac{B}{V}+\frac{C}{V^2}$

$$p=1.013\times10^6\,\mathrm{Pa},T=473.15K$$

$$R=8.314\times10^6(cm^3\cdot Pa)/(K\cdot mol)$$

$$\frac{pV}{R\times T}-\frac{B}{V}-\frac{V}{V^2}-1=0$$

- 用fzero求解?

例题

```
function weili(x)  #定义维利方程的函数
    p=1.013e6;R=8.314e6;T=473.15;B=-388;C=-26000;
    f=p*x/(R*T)-B/x-C/(x^2)-1;
    return f
end
---------
p=1.013e6;R=8.314e6;T=473.15;B=-388;C=-26000;
x0=RT/p #x0=3883.3
x=fzero(weili,x0) #调用此函数
结果x =   3436.2605

或者----------------------------------------
p=1.013e6;R=8.314e6;T=473.15;B=-388;C=-26000;
x0=R*T/p #给出初值
f=x-> p*x/(R*T)-B/x-C/(x^2)-1  #定义维利方程的函数
x=fzero(f,x0) #调用此函数
结果：x=3436.2605cm³
----------------------------------------------------
注意：两者调用形式的不同  思考：初值不同对结果的影响?
```

非线性方程组

化工生产中的非线性方程组问题

例：在合成氨生产中,烃类蒸气发生以下转化反应:

$$CH_4 + H_2O_{(g)} \Leftrightarrow CO + 3H_2$$
$$CO + H_2O_{(g)} \Leftrightarrow CO_2 + H_2$$

已知进料甲烷为1mol,水蒸汽为5mol,反应后总压P=1atm,反应平衡常数为:

$$K_{P1} = \frac{P_{CO}P_{H_2O}^3}{P_{CH_4}P_{H_2O}} = 0.9618$$

$$K_{P2} = \frac{P_{H_2}P_{CO_2}}{P_{CH_4}P_{H_2O}} = 2.7$$

试求反应平衡时各组分的浓度。

解:设反应平衡时有x摩尔甲烷转化成CO,同时生成的CO中又有y摩尔转化成CO_2,则反应平衡时各组分的摩尔数及分压如下:

将平衡时各组分的分压表达式代入反应平衡常数K_{P1}及K_{P2}的表达式得:

化工生产中的非线性方程组问题

组分名称	摩尔数	分压
CH_4	$1-x$	$P_{CH_4} = \frac{1-x}{6+2x}P$
H_2O	$5-x-y$	$P_{H_2O} = \frac{5-x-y}{6+2x}P$
CO	$x-y$	$P_{CO} = \frac{x-y}{6+2x}P$
CO_2	y	$P_{CO_2} = \frac{y}{6+2x}P$
H_2	$3x+y$	$P_{H_2} = \frac{3x+y}{6+2x}P$
总摩尔数	$6+2x$	

$$K_{P1} = \frac{P_{CO}P_{H_2O}^3}{P_{CH_4}P_{H_2O}} = 0.9618 \qquad \frac{(x-y)(3x+y)^3}{(1-x)(5-x-y)(6+2x)^2} = 0.9618$$

$$K_{P2} = \frac{P_{H_2}P_{CO_2}}{P_{CH_4}P_{H_2O}} = 2.7 \qquad \frac{y(3x+y)}{(x-y)(5-x-y)} = 2.7$$

非线性方程组求解

- 直接迭代法: $x = 1 - \frac{(x-y)(3x+y)^3}{(5-x)(6+2x)^2}$

  $y = \frac{2..7(x-y)(5-x-y)}{3x+y}$
- 松弛迭代法: 与松弛因子ω取值有关
- 二分法
- 牛顿法
- 正割法

...

Mworks求解非线性方程组：fsolve

- 求解非线性方程组的函数：fsolve
- 函数库: TyOptimization
- 应用了Levenberg-Marquardt 和信赖域方法基于非线性最小二乘算法
- x,fval,exitflag,output,jacobian = fsolve(fun,x0)
- 从 x0 开始，尝试求解方程 fun(x) = 0（全零数组）。
- fun:多变量函数，写成数组
- x0: 初值，选择重要
- x - 方程的解
- fval - 目标函数 fun 在解 x 处的值
- exitflag - 描述 fsolve 的退出条件的值 exitflag
- output - 提供优化过程信息的结构体
- jacobian - fun 在解 x 处的雅可比矩阵

Mworks求解非线性方程组：fsolve

- x = fsolve(myfun,x0)
- 其中 myfun 是一个 Syslab 函数，定义如下：
- ----------
- function myfun(x)
- F=...
- return F
- end
- ----------
- myfun = x-> begin
- F = ... ## 计算 x 处的函数值
- return F
- end
- ----------
- fun 也可以是匿名函数的函数句柄，直接被调用。
- x, = fsolve(x->F,x0)

Mworks求解非线性方程组：

- 例题：用fsolve求解非线性方程组

$$2x_1 - x_2 = e^{-x_1}$$
$$-x_1 + x_1x_2 = e^{-x_2}$$

- 变换一下成 $f(x)=0$的形式

$$2x_1 - x_2 - e^{-x_1} = 0$$
$$-x_1 + x_1x_2 - e^{-x_2} = 0$$

- ----------
- #定义方程组函数（形式1）
- function demofun(x)
- F=[2*x[1]-x[2]-exp(-x[1]); -x[1]+x[1]*x[2]-exp(-x[2])];
- return F
- end
- #调用函数
- >> x0=[-5 -5];
- >> x=fsolve(demofun,x0)
- 结果：x = 0.8659 1.3112

Mworks求解非线性方程组

- #定义方程组函数（形式2）
- myfun=x->begin #定义函数
- F=[2*x[1]-x[2]-exp(-x[1]); -x[1]+x[1]*x[2]-exp(-x[2])];
- return F
- end
- #调用函数
- x0=[-5,-5];
- x=fsolve(myfun,x0)
- 结果：x = 0.8659 1.3112
- ----------
- #直接调用匿名的函数（形式3）
- x=fsolve(x->[2*x[1]-x[2]-exp(-x[1]); -x[1]+x[1]*x[2]-exp(-x[2])],x0)
- 结果：x = 0.8659 1.3112

$$2x_1 - x_2 - e^{-x_1} = 0$$
$$-x_1 + x_1x_2 - e^{-x_2} = 0$$

非线性方程组求解

练习：对合成氨例题中的非线性方程组用Matlab的fsolve求解

令x=x1, y=x2，将上面的非线性方程组写成f(x)=0的形式

$$\frac{(x-y)(3x+y)^3}{(1-x)(5-x-y)(6+2x)^2}=0.9618$$

$$\frac{y(3x+y)}{(x-y)(5-x-y)}=2.7$$

$$\frac{(x_1-x_2)(3x_1+x_2)^3}{(1-x_1)(5-x_1-x_2)(6+2x_1)^2}-0.9618=0$$

$$\frac{x_2(3x_1+x_2)}{(x_1-x_2)(5-x_1-x_2)}-2.7=0$$

非线性方程组求解

练习用mworks的fsolve命令求解该非线性方程组

$$\begin{cases}\sin x+y^2+\ln z=7\\ 3x+2^y-z^3+1=0\\ x+y+z=5\end{cases}$$

微分方程组的求解

微分方程的概念

- 微分方程：含有自变量、未知函数及其导数的方程
- 常微分方程：未知函数只含有一个变量的微分方程
  Ordinary Differential Equations，ODE
- 偏微分方程：未知函数中含有两个或两个以上变量的微分方程,方程中含有偏导数 Partial Differential Equations，PDE
- 微分方程的阶: 微分方程中最高的导数阶次 >2—高阶微分方程
- 线性与非线性：系数是未知函数的函数
- 微分方程求得的结果往往有很多解（通解），为了得到特定问题的解，必须给出附加条件。
  - 给定微分方程及其初始条件，称为初值问题；
  - 给定微分方程及其边界条件，称为边值问题。

微分方程的概念

- 在自然科学及工程技术领域内的许多问题，都可以用微分方程来描述。
- 在建立微分方程模型时，注意初始和边界条件，这样才是一个完整的定解问题
- 实际应用中只有一些特殊形式的方程，才能找到它的解析解；对于大多数的微分方程无法确定其解析解，只能通过逼近的方法求其数值解。
  - 解析解：具有n阶导数的连续函数（族）
  - 数值解：确定微分方程的解在某些特定自变量处的取值或近似的一组离散的数据。

化工中的微分方程问题

- 化学反应—— 反应速率与物质的量
- 质量传递—— 扩散、对流方程
- 热量传递—— 传导、对流、辐射
- 动量传递—— 强制流动

微分方程：浓度分布、温 度 场、势压梯度

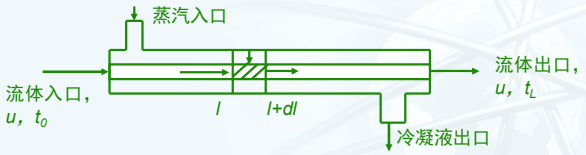
- 间歇反应器的计算
- 活塞流反应器的计算
- 全混流反应器的动态模拟
- 定态一维热传导问题
- 逆流壁冷式固定床反应器一维模型
- 固定床反应器的分散模型

- 集中参数模型：
  - 因变量不随空间坐标变化，模型中自变量仅为时间—常微分方程
- 分布参数模型：
  - 因变量不仅与时间有关，而且与空间有关—偏微分方程（传热、扩散）

化工中的微分方程问题

例：套管式换热器温度分布

- 套管内侧为液体，其温度只随套管的长度改变而变化，忽略径温度变化
- 套管环隙为蒸汽，温度在任何位置均为恒定值，可认为是饱和蒸汽的温度
- 忽略套管内侧流体的纵向热传导
- 在整个套管长度方向上，总传热系数K不变



取一微元进行热量衡算： 流入的热量+传入的热量-流出的热量=0

$$C_p\pi r^2 u\rho t + K\times 2\pi r dl\times(T_W - t) - C_P\pi r^2 u\rho(t+dl\frac{dt}{dl}) = 0$$

化工中的微分方程问题

化简得微分方程：

$$\frac{dt}{dl} = \frac{2K}{u\rho\ C_P r}(T_W - t)$$

微分方程中各变量的含义如下：

t,套管内某一点的温度,K
l,流体在套管内所处的位 置,m
T_W,套管的管壁温度,K
u,套管内流体的速度,m/s
C_p,套管内流体的比热,$J/Kg\bullet K$
ρ 套管内流体的密度,Kg/m^3
r,内套管半径，m

- 通过求解微分方程，可得到管内流体的温度随管子长度而改变的曲线，为化工模拟和设计提供依据
- 如果方程中传热系数、物流性质不随温度或位置的变化，此方程可以解析求解，得到常见的换热器传热方程
- 如果上述性质随温度或位置而改变，此方程只能用数值的方法求解
- 事实上传热系数，物流性质都会随着温度的改变而改变，故在深入研究换热器各点温度分布时，应采用微分方程的数值求解

化工中的微分方程问题

例：物料冷却过程的数学模型

$$\frac{dT}{dt}=-k(T-T_0)$$

它含有自变量t（时间）、未知函数 T（随时间变化的物料温度）、T_0（环境温度）、k（降温速率）以及温度的一阶导数 $\frac{dT}{dt}$,是一个常微分方程。

常微分方程问题求解

初值问题： $\begin{cases} y'=f(x,y) \\ y(a)=y_0 \end{cases}$

- 附加条件在自变量的一端
- 初值问题的数值解法一般采用步进法
  - 欧拉法 Euler
  - 龙格 - 库塔法 Runge-Kutta
- 求 y(x) 在求解区间[a,b] 上各个分点序列x_n（n =1,2,...,m）的数值解 y_n

边值问题： $\begin{cases} y'=f(x,y) \\ y(a)=y_1, y(b)=y_2 \end{cases}$

- 在自变量两端均给定附加条件
- 边值问题可能有解、也可能无解，可能有唯一解、也可能有无数解
- 边值问题数值解法
  - 迭加法
  - 打靶法
  - 松弛法

微分方程的数值求解

- 求微分方程数值解的一般步骤：
  - 区域剖分：首先按一定规则将整个定义域分成若干小块
  - 微分方程离散：构造离散点或片的函数值递推公式或方程
  - 初始、边界条件离散：根据递推公式，将初值或边界值离散化，补充方程，启动递推运算
  - 数值解计算：求解离散系统问题
- 微分方程的定解问题 ⇒ 离散系统的求解问题

常微分方程初值问题求解

常微分方程初值问题求解：欧拉法

$$\begin{cases} y'(x) = f(x,y) \\ y(a) = y_0 \end{cases}, (a \le x \le b)$$

$y'(x)=f(x,y)$

在求解区间[a,b]上作等距分割，步长 $h = \frac{b-a}{m}$，$x_n = x_{n-1} + h$，（n=1-m）

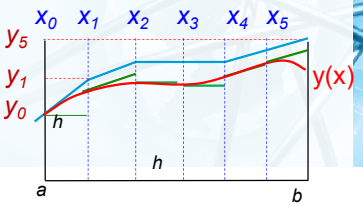
差商近似导数：做 $y'(x)$ 在 $x = x_0$ 处的一阶向前差商 $y'(x_0) \approx \frac{y(x_1) - y(x_0)}{h}$

得 $y(x_1)$ 近似值 y_1：$y_1 = y_0 + hf(x_0, y_0)$

由 $y'(x_n) \approx \frac{y(x_{n+1}) - y(x_n)}{h}$

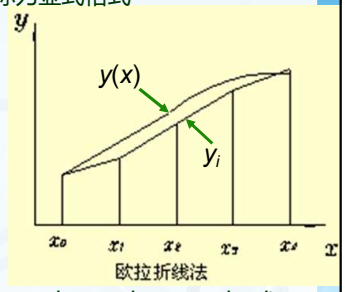
得到计算近似值的向前欧拉公式：

$$y_{n+1} = y_n + hf(x_n, y_n)$$



向前欧拉公式 Explicit Euler Formula

- 由 y_n 直接算出 y_{n+1} 值的计算格式称为显式格式
- 向前欧拉公式是显式格式。
- 欧拉方法的几何意义：
  - 以 y_0' 作为斜率，通过点 (x_0, y_0) 做一条直线，它与直线 $x=x_1$ 的交点就是 y_1。
  - y_{n+1} 是以 y_n' 作为斜率，经过点 (x_n, y_n) 的直线与直线 $x=x_{n+1}$ 的交点。
- 欧拉法也称为欧拉折线法(Euler's polygonal arc method)

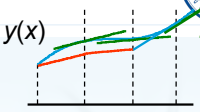


其他欧拉公式

向后欧拉公式（隐式）Implicit Euler Formula

做y在x_1处的一阶向后差商式 $y'(x_1) \approx \frac{y(x_1) - y(x_0)}{h}$

$$y_1 = y_0 + hf(x_1, y_1) \quad y_{n+1} = y_n + hf(x_{n+1}, y_{n+1})$$



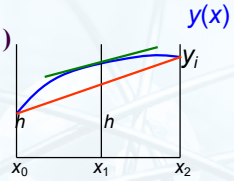
中心欧拉公式 midpoint Euler formula：

做y在x_1处的中心差商

$y'(x)=f(x,y)$

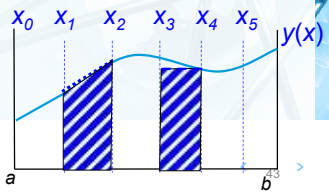
$$y'(x_1) \approx \frac{y(x_2) - y(x_0)}{2h}$$

$$y_{n+1} = y_{n-1} + 2hf(x_n, y_n)$$



梯形公式 Trapezoid Formula

$$y_{n+1} = y_n + \frac{h}{2}(f(x_n, y_n) + f(x_{n+1}, y_{n+1}))$$



欧拉公式比较

方 法	👍	👎
显式欧拉	简单	精度低
隐式欧拉	稳定性最好	精度低，计算量大
梯形公式	精度提高	计算量大
中心公式	精度提高，显式	多一个初值，可能影响精度

Can't you give me a formula with all the advantages yet without any of the disadvantages?

龙格 - 库塔法　Runge-Kutta Method

- 优点:
  - 计算精确度较高，能满足通常的计算要求;
  - 容易编制程序;
  - 每次计算$y(x_{n+1})$，只用到前一步的计算结果y_n，因此，在已知初始值y_0的条件下，就可自动地进行计算，是单步法，而且计算过程中可随时改变步长。
- 缺点:
  - 每前进一步需要计算多次f(x,y)的值，因此，计算工作量较大，且其截断误差难以估计。
- 在实际应用上，一般当要求更高精确度时，采用的办法是缩小步长，而不是采用更高阶的公式，因为高阶公式的计算太复杂，一般选用标准四阶龙格-库塔方法即可。

Mworks求解常微分方程初值问题

- Syalab 中的常微分方程 (ODE) 求解器可对具有各种属性的初始值问题进行求解。求解器可以处理刚性或非刚性问题、具有质量矩阵的问题、微分代数方程 (DAE) 或完全隐式问题。
- ode23s 求解器只能解算质量矩阵为常量的问题。ode15s 和 ode23t 可以解算具有奇异质量矩阵的问题，称为微分代数方程 (DAE)。
- ode45 是一个通用型 ODE 求解器，是解算大多数问题时的首选。

函数名	简介
ode45	求解非刚性微分方程 - 中阶方法
ode23	求解非刚性微分方程 - 低阶方法
ode78	求解非刚性微分方程 - 高阶方法
ode89	求解非刚性微分方程 - 高阶方法
ode113	求解非刚性微分方程 - 变阶方法

函数名	简介
ode15s	求解刚性微分方程和 DAE - 变阶方法
ode23s	求解刚性微分方程 - 低阶方法
ode23t	求解中等刚性的 ODE 和 DAE - 梯形法则
ode23tb	求解刚性微分方程 - 梯形法则 + 后向差分公式

函数名	简介
ode15i	解算全隐式微分方程 - 变阶方法
decic	为 ode15i 计算一致的初始条件

函数名	简介
odeget	提取 ODE 选项值
odeset	为 ODE 和 PDE 求解器创建或修改 options 结构体

求解器	特点	说明
ode45	一步算法，4,5 阶 Runge-Kutta 方法累积截断误差 $(\Delta x)^3$	大部分场合的首选算法
ode23	一步算法，2,3 阶 Runge-Kutta 方法累积截断误差 $(\Delta x)^3$	使用于精度较低的情形
ode113	多步法，Adams 算法，高低精度均可达到 $10^{-3}\sim10^{-6}$	计算时间比 ode45 短
ode23t	采用梯形算法	适度刚性情形
ode15s	多步法，Gear's 反向数值积分，精度中等	若 ode45 失效时，可尝试使用
ode23s	一步法，2 阶 Rosebrock 算法，低精度。	当精度较低时，计算时间比 ode15s 短

Mworks求解常微分方程初值问题:

ode45:求解非刚性微分方程 - 中阶方法

函数库: TyMath

t,y, = ode45(odefun,tspan,y0)

$$\begin{cases} y' = f(x,y) \\ y(a) = y_0 \\ x \in (a,b) \end{cases}$$

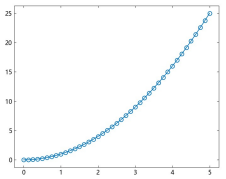
- 其中 tspan = [t0 tf]）求微分方程组 从 t0 到 tf 的积分，初始条件为 y0。解数组 y 中的每一行都与列向量 t 中返回的值相对应。
- 所有 ODE 求解器都可以解算 y'=f(t,y) 形式的方程组，或涉及质量矩阵 M(t,y)y'=f(t,y) 的问题。求解器都使用类似的语法。

Mworks求解常微分方程初值问题方法：ODEfun

- ODEfun函数一般作为整个求解程序的一个子函数，表示ode求解问题
- f - 常微分方程形式的函数，表示dy/dt
- 对于标量 t 和列向量 y 来说，函数 odefun = (t,y) -> dydt 必须返回列向量 dydt，该列向量对应于 f(t,y)。
- odefun 必须有一个自变量t和函数y作为输入变量，一个y的导数(dy/dt)作为输出变量，即使其中一个参数未在函数中使用也是如此。
- 其中自变量t不论在odefun函数中是否使用都必须作为第一输入变量，y则必须作为第二输入变量，位置不能颠倒。
- 对于方程组，odefun 的输出为向量。向量中的每个元素是一个方程的解。

Mworks求解常微分方程初值问题方法：ODEfun

- 在对求解器的调用中
- (1)可将只有一个解分量的简单 ODE 指定为匿名函数。该函数必须同时接受两个输入 (t,y)，即使在该函数中一个输入未使用也是如此。
- 如 $y' = 2t$
- 指定时间区间 [0 5] 和初始条件 y(t=0) = 0。
- f=(t,y)->2*t
- t,y, = ode45(f, [0 5], 0);
- 或 t,y, = ode45((t,y)->2*t, [0 5], 0)
- plot(t,y,"-o")



Mworks求解常微分方程初值问题方法：ODEfun

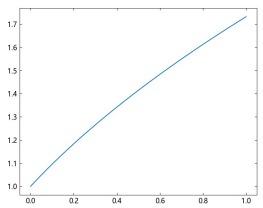
- (2) 也可以定义一个odefun函数(形式1)
- odefun函数：
- funtion odefun(t,y)
  dydt=f;
  return dydt
  end

ode输入函数

自变量在前因变量在后

输出变量为因变量导数的表达式

$$\begin{cases} y' = y - \dfrac{2x}{y} \\ y(0) = 1 \end{cases} \quad (0 \le x \le 1)$$



- 例：求解初值问题
- odefunction函数：
- function fun(x,y)
- dydx=y-2*x/y;
- return dydx
- end
- 调用：
- x,y,=ode45(fun,[0 1],1);
- plot(x,y)

Mworks求解常微分方程初值问题方法：ODEfun

- (2) 也可以定义一个odefun函数（形式2）
- function odefun(t,y)
- return f #(f=dydt)
- end

例如，要解算 $y' = 5y - 3$，请使用此函数：

```
function odefun(t,y)
    return 5*y .- 3
end
```

- 例：
- function fun1(x,y)
- return y-2*x/y;
- end
- 调用：
- x,y,=ode45(fun1,[0 1],1);
- plot(x,y)

$$\begin{cases} y' = y - \dfrac{2x}{y} \\ y(0) = 1 \end{cases} \quad (0 \le x \le 1)$$

Mworks求解常微分方程初值问题

- 课堂练习：

$$\begin{cases} y' = y + y^2 \\ y(0) = 1 \end{cases} \quad (0 \le x \le 1)$$

- function fun(x,y)
- f=y+y^2;
- return f
- end
- .....

常微分方程组的数值解法

将由m个一阶方程组成的常微分方程初值问题写成向量形式：

$$\begin{cases} \frac{dy_1}{dt} = f_1(t, y_1, y_2, \dots, y_m) \\ \frac{dy_2}{dt} = f_2(t, y_1, y_2, \dots, y_m) \\ \dots \\ \frac{dy_m}{dt} = f_m(t, y_1, y_2, \dots, y_m) \\ y_1(a) = \eta_1 \\ y_2(a) = \eta_2 \\ \dots \\ y_m(a) = \eta_m \end{cases}$$

其中 $(a \le t \le b)$

$$\begin{cases} \frac{dY}{dt} = F(t, y) \\ Y(a) = \eta \end{cases}$$

$$F(t,y) = \begin{pmatrix} f_1(t, y_1, \dots, y_m) \\ f_2(t, y_1, \dots, y_m) \\ \dots \\ f_m(t, y_1, \dots, y_m) \end{pmatrix}$$

$$Y(t) = \begin{pmatrix} y_1(t) \\ y_2(t) \\ \dots \\ y_m(t) \end{pmatrix} \qquad \eta = \begin{pmatrix} \eta_1 \\ \eta_2 \\ \dots \\ \eta_m \end{pmatrix}$$

Mworks求解常微分方程组初值问题

- 对于方程组，odefun 的输出为向量。向量中的每个元素是一个方程的解。
- odefun函数：
- funtion odefun(t,y)
  dydt=[f1; f2; .....];
  return dydt
  end

tspan=[t1 t2];
y0=[y1 y2 ...];
t,y, = ode45(odefun,tspan,y0);
plot(t,y[:,1],t,y[:,2], ...)

$y_1' = y_1 + 2y_2$

$y_2' = 3y_1 + 2y_2$

使用函数：

```
function odefun(t,y)
    dydt = [y[1]+2*y[2];3*y[1]+2*y[2]]
    return dydt
end
```

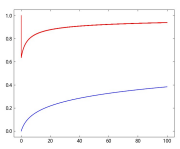

Mworks求解一阶常微分方程组初值问题

常微分方程组与单个常微分方程求解方法相同，只需在编写odefile时将整个方程组作为一个向量输出。

$$\begin{cases} y_1' = 0.04(1 - y_1) - (1 - y_2)y_1 + 0.0001(1 - y_2)^2 \\ y_2' = -10^4 y_1' + 3000(1 - y_2)^2 \\ y_1(0) = 0, y_2(0) = 1, \quad 0 \le x \le 100 \end{cases}$$

```
%---odefun----------------
function myfun(x,y)
dy1dx = 0.04*(1-y[1])-(1-y[2]).*y[1]+0.0001*(1-y[2]).^2;
dy2dx = -1e4*dy1dx + 3000*(1-y[2]).^2;
f = [dy1dx; dy2dx];
return f
end
```

```
%-----solve ode----------------
x,y,=ode45(myfun, [0 100],[0 1]);
plot(x,y[:,1],"b",x,y[:,2],"r")
```



练习题

- 某间歇式反应器中进行反应 A→ B→ C，设c_A、c_B分别为A、B的浓度，方程如下：

$$\begin{cases}\dfrac{dc_A}{dt}=-4000\cdot\exp(-2500/T)\cdot c_A^2\\ \dfrac{dc_B}{dt}=4000\cdot\exp(-2500/T)\cdot c_A^2-620000\cdot\exp(-5000/T)\cdot c_B\\ c_A(0)=1, c_B(0)=0\end{cases}$$

- 当T=389K时，用Matlab的ode45命令求C_A和C_B随时间变化的值，时间自己定，如可以从0-20/50，并作图

常微分方程边值问题求解（了解）

Mworks求解边值问题

- 边界值问题 (BVP) 是受限于边界条件的常微分方程。与初始值问题不同，BVP 可以有一个有限解、无解或有无限多个解。
- 解的初始估计值是求解 BVP 必不可少的一部分，估计值的质量对于求解器性能乃至计算成功与否都至关重要。
- bvp4c 和 bvp5c 求解器适用于具有两点边界条件、多点条件、奇异值或未知参数的边界值问题。

▾ 求解器

bvp4c	求解边界值问题 - 四阶方法
bvp5c	求解边界值问题 - 五阶方法
bvpinit	得出边界值问题求解器的初始估计值

▾ Get/Set 选项

bvpget	提取使用 bvpset 创建的 options 结构体中的属性
bvpset	创建或更改边界值问题的 options 结构体

▾ 计算和扩展解

deval	计算微分方程解结构体
bvpxtend	构造用于扩展边界值解的估计值结构体

59

Mworks求解边值问题

- bvp4c：求解边界值问题 - 四阶方法
- 函数库: TyMath
- sol = bvp4c(odefun,bcfun,solinit)
- 使用 bcfun 描述的边界条件和初始解估计值 solinit 对 odefun 指定的形式为 y′ = f(x,y) 的微分方程组进行积分。使用 bvpinit 函数创建初始估计值 solinit，该函数还定义了要强制应用 bcfun 中边界条件的点。

- bcfun：边界条件，指定为计算边界条件中残差的函数。odefun 和 bcfun 必须接受相同数量的输入参数。
- 要编写 bcfun 代码，请对列向量 ya 和 yb 使用函数 bcfun(ya,yb)。
- 例如，如果 y[a] = 1 且 y[b] = 0，则边界条件函数为

```
function bcfun(ya,yb)
    return [ya[1]-1; yb[1]]
end
```

- solinit： 解的初始估计值，结构体。使用 bvpinit 创建 solinit。
- 与初始值问题不同，边界值问题可以无解、有有限数量的解或者有无限数量的解。求解 BVP 过程的重要部分是提供所需解的估计值。估计值的准确与否对求解器性能甚至是能否成功计算来说至关重要。有关创建良好初始估计值的一些指导原则，请参阅解的初始估计值。

- bvpinit：得出边界值问题求解器的初始估计值
- solinit = bvpinit(x,yinit)
- 使用初始网格 x 和初始解估计值 yinit 来得出边界值问题解的初始估计值。然后，可以使用初始估计值 solinit 作为 bvp4c 或 bvp5c 的输入之一来求解边界值问题。创建 BVP 的解的初始估计值，用 bvp4c 求解 BVP，然后将解扩展到新域。
- 得出 BVP 问题的解的良好初始估计值可能是求解问题最困难的部分。BVP 解不一定唯一，因此初始估计值可以是求解器确定要返回的解的决定因素。初始估计值应该满足边界条件，其间的行为应该反映您对此问题的一般期望（解是否振荡，是否为简单的线性函数等）。

60

小结

- 学会用Mworks求解化工中的非线性方程问题
- 学会用Mworks求解化工中的常微分方程组初值问题
- 自学用Mworks求解化工中的常微分方程组边值问题