Cheatography

Numerical Methods 262 Cheat Sheet

by Kamva via cheatography.com/169429/cs/35481/

Curve fitting: Higher order Polynomials

The formula: ax + by + cz = 1

```
A= [-1 -1 -3;1 0 0;2 1
2;2 3 6];
b = [1;1;1;1];
x = A\b;
```

NOTE: A**x** = b...**backslash(\)** to solve for **a**,**b**,**c**

Curve fitting: linearization

Exponential linearized function

The formula is $y = be^{mx}$:

```
x = [1 2 3 4]; y = [6 3
2 1]; Given
a = polyfi t(x ,lo -
g(y),1)
>>a = -0.5781 2.3411;
m = a(1)
>>m = -0.5781;
b = exp(a(2));
>>b = 10.3923;
```

The final product: $y = 10.39e^{-0.5781x}$

Eigenvalues & Eigenvectors

```
A=[2 -1 0;-1 2 -1;0 -1
2]; a=eig(A);
[V,D] = eig(A) V - eigenv-
ectors
V(:,1):get 1st column
v1 = V(:,1);
v1 = v1/v1(1);: normalise
```

eigenvector

ODEs using ode45

```
a = 0.1; b = 0.002; c =
0.0025; d = 0.2;
f = @(t,x) [-
ax(1)+bx(1)*x(2);
dx(2)-cx(1)*x (2)];
X0 = [20; 100];
Tspan = linspa ce( 0,1 -
50, 1000);
[T,X] = ode45( f,T spa -
n,X0);
plot(T ,X, 'Li neW idt -
h',2);
```

ODEs using ode45

```
a = 0.1; b = 0.002; c =
0.0025; d = 0.2;
f = @(t,x) [-
ax(1)+bx(1)*x(2);
dx(2)-cx(1)*x (2)];
X0 = [20; 100];
Tspan = linspa ce( 0,1 -
50, 1000);
[T,X] = ode45( f,T spa -
n,X0);
plot(T ,X, 'Li neW idt -
h',2);
```

Gauss Integration

Gauss 2-pt rule | Gauss 3-pt rule

```
f=@(x) 1/2*sq rt( 16+ -
(x+ 3).^ 4)./( x+3)^2;
G2=f(- 1/s qrt (3) )+f -
(1/ sqr t(3));
>>G2= 1.1289;
```

and

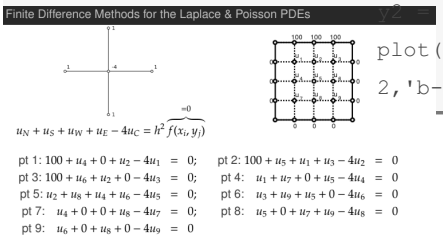
```
G3=5/9f(-sqr -
t(3 /5) )+8/9f(0)+5 /9
* f(s qrt (3/5));
>>G3= 1.1319;
```

ODEs: Boundary-value problems (BVPs)

Finite Difference Method

```
y0=1;y1=2; % N = 1/h;h =
0.2
i = [1:N-1]';
xi = i*h; % Interior
nodes
A=(h^2 -2) *ey e(N-1); %
Diagonal part of A
A=A+di ag( 1-
0.5hxi(2:N -1) ,-1);%
Add sub- and superd -
iagonal
A=A+di ag( -
1+0.5hxi(1:N -2) ,+1);%
Add sub- and superd -
iagonal
b = 2 h^2xi;% RHS refined
formula
b(1)=b (1) -(1-
0.5hxi(1)) *y0;% Add
boundary contri butions
b(N-1) =b( N-1 )-( -
1+0.5hxi(N-1 ))*y1;% Add
boundary contri butions
y = sparse (A)\b; %
Solve
```

Finite Difference: Laplace&-Poisson PDEs



You then use backslash after you have the equations

Multiple Integrals

```
f = @(x,y) x.^2-y.^2-1;
a = 0; b = 3; g = 0; p =
1;
integr al2 (f, -
a,b ,g,p);
>>ans = 5.0000 000 000 -
01309;
```

or

```
f = @(x,y) y;
a = 0; b = pi/4;
g = @(x) sin(x); p =
@(x) cos(x);
m = integr al2 (f, -
a,b ,g,p);
>>m = 0.2500 000 000 -
00013;
```

Curve fitting& Interpolation

Least Squares: Backslash & Polyfit

y = a0 + a1x + a2x^2:

```
x = [1; 2; 3; 4]; y =
[6; 3; 2; 1];
A = [ones(4,1) x x.^2];
a = A\y
a = 9.5000 -4.1000
0.5000
a2 = polyfi t(x ,y,2)
xx = linspa ce( 0,5);
y2 = polyva l(a 2,xx);
plot(x ,y, 'r.' ,x x,y -
2,'b-')
```

Error,Residual,Norms,Condition number

```
x = [1; -1; 3; -5];
norm(x ,inf)||x||inf(sub-script)
```

||A||2 = (||A**x**||2)/(||**x**||2)

~ cond(A) = (||A||)(||A^{-1}||)

~`cond(A)` >> 1:ill-condition-ed&close singular matrix

~`cond(A)` ≈ 1:well-conditioned

~ decimals digits that trustworthy = $10^{-a} = 10^{c} \times 10^{-d}$

a = d-c

~r = b - A**x**

det (A) = 0; No A^{-1}; Singular matrix

test singularity: `cond,c ond - est ,rank`

Built-in Functions

Integral	Trapz

```
f=@(x) sqr t(1 +x.^ -4)./x.^2;
L=inte gra l(f ,1,2);
>>L =1.1 32 090 393 -305918;
```

or

```
y=[0 0.3 0.5 1 1.5 2 2.5 3 4 5];
v=[0 0.4 0.5 0.56 0.6 0.63 0.66 0.68 0.71 0.72];
Q=10*t rap z(y,v);
>>Q =30.8000;
```

Euler's Method

Euler's Method:

$$\frac{dy}{dx} = f(x, y), \quad y(x_1) = y_1 \quad \Rightarrow \quad y_{i+1} = y_i + h\, f(x_i, y_i), \quad i = 1, 2, \ldots$$

```
f=@(x,y) -x*y; % Define rhs
N=100; % Number of steps
a=1; b=2; % Interval
h=(b-a)/N; % step size
x(1)=1; y(1)=4; % Initial values
for i=1:N
y(i+1) =y( i)+ h*f (x( -i), y(i)); % Euler
x(i+1) =x( i)+h;
end
```

Runge-Kutta Methods

```
f = @(x,y) -x*y; %Define rhs
N = 100; % Number of steps
....
for i = 1:N
K1 = f(x(i) ,y(i));
K2 = f(x(i)+0.5h,y(i)+0.5h*K1);
K3 = f(x(i)+0.5h,y(i)+0.5h*K2);
K4 = f(x(i+ 1), -y(i )+h *K3);
x(i+1) = x(i)+h;
y(i+1) = y(i)+ (1/6)h(K1+2K 2+2 K3+K4);
end
```

Runge-Kutta Methods

```
f = @(x,y) -x*y; %Define rhs
N = 100; % Number of steps
....
for i = 1:N
K1 = f(x(i) ,y(i));
K2 = f(x(i)+0.5h,y(i)+0.5h*K1);
K3 = f(x(i)+0.5h,y(i)+0.5h*K2);
K4 = f(x(i+ 1), -y(i )+h *K3);
x(i+1) = x(i)+h;
y(i+1) = y(i)+ (1/6)h(K1+2K 2+2 K3+K4);
end
```

Numerical Differentiation

Tables for first derivatives:

First Derivative		
Method	Formula	Truncation Error
Two-point forward difference	$f'(x_i) = \frac{f(x_{i+1}) - f(x_i)}{h}$	$O(h)$
Three-point forward difference	$f'(x_i) = \frac{-3f(x_i) + 4f(x_{i+1}) - f(x_{i+2})}{2h}$	$O(h^2)$
Two-point backward difference	$f'(x_i) = \frac{f(x_i) - f(x_{i-1})}{h}$	$O(h)$
Three-point backward difference	$f'(x_i) = \frac{f(x_{i-2}) - 4f(x_{i-1}) + 3f(x_i)}{2h}$	$O(h^2)$
Two-point central difference	$f'(x_i) = \frac{f(x_{i+1}) - f(x_{i-1})}{2h}$	$O(h^2)$
Four-point central difference	$f'(x_i) = \frac{f(x_{i-2}) - 8f(x_{i-1}) + 8f(x_{i+1}) - f(x_{i+2})}{12h}$	$O(h^4)$

```
x = 0;
f = @(x) sqrt(1 -2* -sin (x));
h = [0.002 0.001 0.0005 0.00025];
D2f = @(h) (f(x)-2f(x+h) +f(x+2h))./h.^2;
D2f(h) = -1.0040 -1.0020 -1.0010 -1.0005; *error decreases by factor 2 thus n = 1
```

Composite Simpson's Rule

Remember: N even

```
f = @(x) sqrt(1 +x.^ -4)./x.^2;
a = 1; b = 2;
N = 100; h = (b-a)/N;
x = a+[0:N]*h;
fx = f(x);
L = h/3 (fx(1)+4sum(fx -(2: 2:N ))+ 2*s um( -fx( 3:2 :N- 1)) +fx -(N+1))
```

or

```
w = ones(s ize (x));
w(2:2:N) = 4; w(3:2:N-1) = 2;
L = h/3 sum(w.f(x))
>>L = 1.1320 903 946 -95845;
```

Composite Trapezium Rule

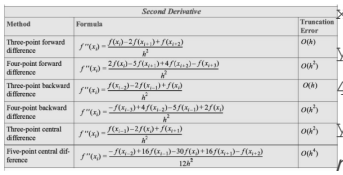
```
f = @(x) sqrt(1 +x.^ -4)./x.^2;
a = 1; b = 2;
N = 100; h = (b-a)/N;
x = a+[0:N]*h;
fx = f(x);
L = h3 ( 0.5fx(1) + sum(fx (2:N)) + 0.5*fx -(N+1) )
```

or

```
w = ones(s ize (x));
w(1) = 0.5; w(N+1) = 0.5;
L = h sum(w.f(x));
>>L = 1.1321 016 727 -88808;
```

Numerical Partial Differentiation

Tables for second derivatives:



Second Derivative		
Method	Formula	Truncation Error
Three-point forward difference	$f''(x_i) = \frac{f(x_i) - 2f(x_{i+1}) + f(x_{i+2})}{h^2}$	$O(h)$
Four-point forward difference	$f''(x_i) = \frac{2f(x_i) - 5f(x_{i+1}) + 4f(x_{i+2}) - f(x_{i+3})}{h^2}$	$O(h^2)$
Three-point backward difference	$f''(x_i) = \frac{f(x_{i-2}) - 2f(x_{i-1}) + f(x_i)}{h^2}$	$O(h)$
Four-point backward difference	$f''(x_i) = \frac{-f(x_{i-3}) + 4f(x_{i-2}) - 5f(x_{i-1}) + 2f(x_i)}{h^2}$	$O(h^2)$
Three-point central difference	$f''(x_i) = \frac{f(x_{i-1}) - 2f(x_i) + f(x_{i+1})}{h^2}$	$O(h^2)$
Five-point central difference	$f''(x_i) = \frac{-f(x_{i-2}) + 16f(x_{i-1}) - 30f(x_i) + 16f(x_{i+1}) - f(x_{i+2})}{12h^2}$	$O(h^4)$

```
x = 0; y = 0;
f = @(x,y) sqrt(1+2 x-
3y.^2);
h = [0.2 0.1 0.05
0.025];
Lapf = @(h) (f(x+h ,y) -
+f( x-h ,y) +f( x,y -
+h) +f( x,y -h) -4* -
f(x ,y) )./ h.^2;
Lapf(h) = -4.1505 -
4.0356 -4.0088 -4.0022;
```

Spline

Extrapolate data set

```
x = [22 35 48 61 74];
y = [320 490 540 500
480];
yy = spline (x, y,42);
```

not-a-knot

```
>>yy = 531.1971;
xx = linspa ce( 20, -
80, 1000);
```

*clamped conditions: `[0 y 0]`

```
ss = spline (x, y,xx);
plot(x ,y, 'r.' ,x x,s -
s,' b-');
```

Modified Euler

Implicit Trapezium Rule

$$y_{i+1} = y_i + \frac{1}{2}h\left(f(x_i, y_i) + f(x_{i+1}, y_{i+1})\right)$$

```
f = @(x,y) -x*y; %
Define rhs
N = 100; % Number of
steps
a = 1; b = 2; % Interval
h= (b-a)/N; % step size
x(1)=1 ;y(1) = 4; %
Initial values
for i = 1:N7
yeu = y(i)+h *f( -
x(i ),y (i)); % Euler
x(i+1) = x(i)+h;
y(i+1) =
y(i)+0.5h(f(x(i ),y -
(i) )+f (x( i+1 ),y -
eu));
end
```