

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: Ax = 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 = polyfit(x, log(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 - eigenvectors`
`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 = linspace(0,150,1000);`
`[T,X] = ode45(f,Tspan,X0);`
`plot(T,X,'LineWidth',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 = linspace(0,150,1000);`
`[T,X] = ode45(f,Tspan,X0);`
`plot(T,X,'LineWidth',2);`

Gauss Integration

Gauss 2-pt rule	Gauss 3-pt rule
<code>f=@(x) 1/2*sqrt(16+(x+3).^4)./(x+3)^2;</code>	
<code>G2=f(-1/sqrt(3))+f(1/sqrt(3));</code>	
<code>>>G2= 1.1289;</code>	
and	
<code>G3=5/9*f(-sqrt(3)/5)+8/9*f(0)+5/9*f(sqrt(3)/5);</code>	
<code>>>G3= 1.1319;</code>	

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) *eye(N-1); % Diagonal part of A`
`A=A+diag(1-0.5hxi(2:N-1),-1),-1);% Add sub- and superdiagonal`
`A=A+diag(1+0.5hxi(1:N-2),+1);% Add sub- and superdiagonal`
`b = 2h^2xi;% RHS refined formula`
`b(1)=b(1)-(1-0.5hxi(1))*y0;% Add boundary contributions`
`b(N-1)=b(N-1)-(1+0.5hxi(N-1))*y1;% Add boundary contributions`
`y = sparse(A)\b; % Solve`

Finite Difference: Laplace & Poisson PDEs

Finite Difference Methods for the Laplace & Poisson PDEs

$u_N + u_S + u_W + u_E - 4u_C = h^2 f(x,y)$

pt 1: $100 + u_4 + 0 + u_2 - 4u_1 = 0;$	pt 2: $100 + u_5 + u_1 + u_3 - 4u_2 = 0$
pt 3: $100 + u_6 + u_2 + 0 - 4u_3 = 0;$	pt 4: $u_1 + u_7 + 0 + u_5 - 4u_4 = 0$
pt 5: $u_2 + u_8 + u_4 + u_6 - 4u_5 = 0;$	pt 6: $u_3 + u_9 + u_5 + 0 - 4u_6 = 0$
pt 7: $u_4 + 0 + 0 + u_8 - 4u_7 = 0;$	pt 8: $u_5 + 0 + u_7 + u_9 - 4u_8 = 0$
pt 9: $u_6 + 0 + u_8 + 0 - 4u_9 = 0$	

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 0 1309;`
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 000 000 000 13;`

Curve fitting & Interpolation

Least Squares: Backslash & Polyfit

$y = a_0 + a_1x + a_2x^2$:
`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 0`
`a2 = polyfit(x ,y,2)`
`linspace(0,5);`
`y2 = polyval(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(subscript)
||A||2 = (||Ax||2)/(||x||2)
~ cond(A) = (||A||)(||A^-1||)
~ cond(A) >> 1: ill-conditioned & close singular matrix
~ cond(A) ≈ 1: well-conditioned
~ decimals digits that trustworthy = 10^-a = 10^c x 10^-d
a = d-c
~ r = b - Ax
det(A) = 0; No A^-1; Singular matrix
test singularity: cond, condition, test, rank
```

Built-in Functions

```
Integral Trapz
f=@(x) sqrt(1+x.^4)./x.^2;
L=integral(f,1,2);
>>L = 1.132090393305918;
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*trapz(y,v);
>>Q = 30.8000;
```

Euler's Method

```
Euler's Method:
dy/dx = f(x,y), y(x0) = y0 => y_{i+1} = y_i + h*f(x_i, y_i), i = 1, 2, ...

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+2K2+2K3+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+2K2+2K3+K4);
end
```

Numerical Differentiation

Tables for first derivatives:

Method	Formula	Truncation Error
Two-point forward difference	$f'(x) \approx \frac{f(x_1) - f(x_0)}{h}$	$O(h)$
Three-point forward difference	$f'(x) \approx \frac{-2f(x_0) + 4f(x_1) - f(x_2)}{2h}$	$O(h^2)$
Two-point backward difference	$f'(x) \approx \frac{f(x) - f(x_{-1})}{h}$	$O(h)$
Three-point backward difference	$f'(x) \approx \frac{f(x) - 4f(x_{-1}) + 3f(x_{-2})}{2h}$	$O(h^2)$
Two-point central difference	$f'(x) \approx \frac{f(x_1) - f(x_{-1})}{2h}$	$O(h^2)$
Four-point central difference	$f'(x) \approx \frac{f(x_2) - 8f(x_1) + 8f(x_{-1}) - f(x_{-2})}{12h}$	$O(h^4)$

```
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*sum(fx(3:2:N-1))+fx(N+1))
or
w = ones(size(x)); w(2:2:N) = 4; w(3:2:N-1) = 2;
L = h/3sum(w.*fx)
>>L = 1.132090394695845;
```

Composite Trapezium Rule

```
f = @(x) sqrt(1+x.^4).
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(size(x)); w(1) = 0.5; w(N+1) = 0.5;
L = hsum(w.*fx);
>>L = 1.13210167278808;
```



Numerical Partial Differentiation

Spline

Tables for second derivatives:



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

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 = linspace( 20, 80, -1000);
*clamped conditions: [0 y 0]
ss = spline (x, y,xx);
plot(x ,y, 'r.', x xx, s s, ' b-');
```

```
x = 0; y = 0;
f = @(x,y) sqrt(1+2x-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;
```

Modified Euler

Implicit Trapezium Rule

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

```
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-1
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 ),yeu));
end
```