Math.375
III- Interpolation

Vageli Coutsias

Problems

- Polynomial Interpolation in 1d: VanderMonde and Newton
- Piecewise linear and cubic interpolation in 2d
- Matrices and BW images
- Color Images

```
function a = InterpV(x,y)
% a = InterpV(x,y)
% This computes the Vandermonde polynomial interpolant
% where x is a column n-vector with distinct components
% and y is a column n-vector.
% a is a column n-vector with the property that if
%       p(x) = a(1) + a(2)x + ... + a(n)x^(n-1)
% then
%       p(x(i)) = y(i), i = 1:n
 n = length(x);
 V = ones(n,n);
 for j = 2:n
   % set up column j
   V(:,j) = x.*V(:,j-1);
 end
 a = V\y;
```

```
function pVal = HornerV(a,z)
%    pVal = HornerV(a,z)
% evaluates the VanderMonde interpolant of z where
% a is an n-vector and z is an m-vector.
%
% pVal is a vector the same size as z with the property that if
%
%       p(x) = a(1) + a(2)x + ... + a(n)x^(n-1)
% then
%       pVal(i) = p(z(i)), i = 1:m
n = length(a);
m = length(z);
pVal = a(n)*ones(size(z));
for k = n-1:-1:1
   pVal = z.*pVal + a(k);
end
```

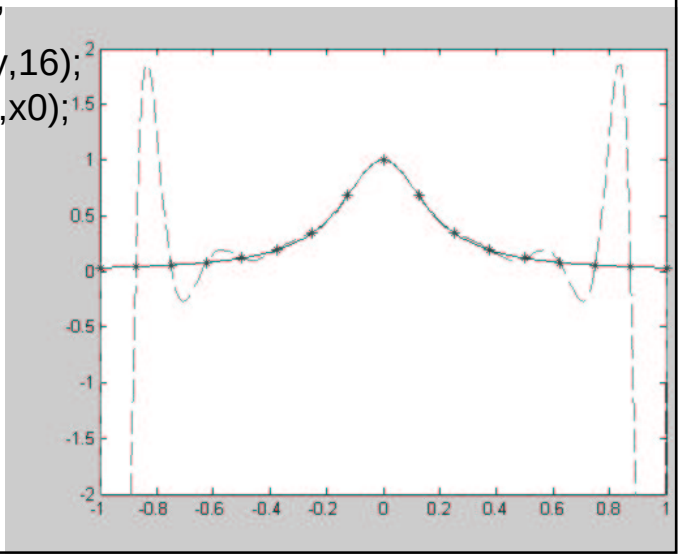
```
function c = InterpNRecur(x,y)
% c = InterpNRecur(x,y)
% The Newton polynomial interpolant.
% x is a column n-vector with distinct components and y is
% a column n-vector. c is  a column n-vector with the property
% that if
%      p(x) = c(1) + c(2)(x-x(1))+...+ c(n)(x-x(1))...(x-x(n-1))
% then
%      p(x(i)) = y(i), i=1:n
n = length(x);
c = zeros(n,1);
c(1) = y(1);
if n > 1
   c(2:n) = InterpNRecur(x(2:n),(y(2:n)-y(1))./(x(2:n)-x(1)));
end
```

```
function c = InterpN(x,y)
% c = InterpN(x,y)
% The Newton polynomial interpolant.
% x is a column n-vector with distinct components and y is
% a column n-vector. c is  a column n-vector with the property
% that if
%      p(x) = c(1) + c(2)(x-x(1))+...+ c(n)(x-x(1))...(x-x(n-1))
% then
%      p(x(i)) = y(i), i=1:n.

n = length(x);
for k = 1:n-1
   y(k+1:n) = (y(k+1:n)-y(k)) ./ (x(k+1:n) - x(k));
end
c = y;
```

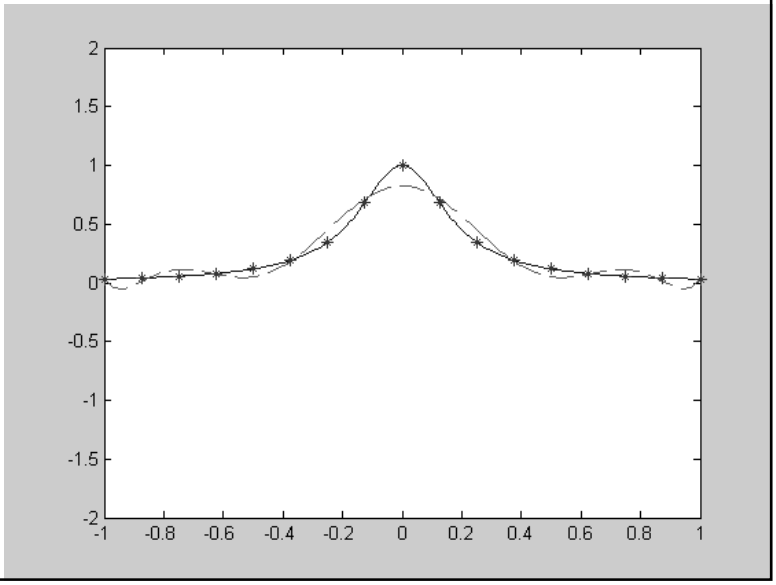
```
function pVal = HornerN(c,x,z)
% pVal = HornerN(c,x,z)
% Evaluates the Newton interpolant on z where
% c and x are n-vectors and z is an m-vector.
%
% pVal is a vector the same size as z with the property
% that if
% p(x) = c(1) +  c(2)(x-x(1))+ ... + c(n)(x-x(1))...(x-x(n-1))
% then
%        pval(i) = p(z(i)) , i=1:m.
n = length(c);
pVal = c(n)*ones(size(z));
for k=n-1:-1:1
   pVal = (z-x(k)).*pVal + c(k);
end
```

9/26/02 7

```
for i = 1:m+1                                   %Equally Spaced
    x(i,1) = .5*(b+a) +.5*(b-a)*(2*(i-1)-m)/m ;
end
y = 1./(1+ll*x.^2);

[P,S] = polyfit(x,y,16);
pVal = polyval(P,x0);
```

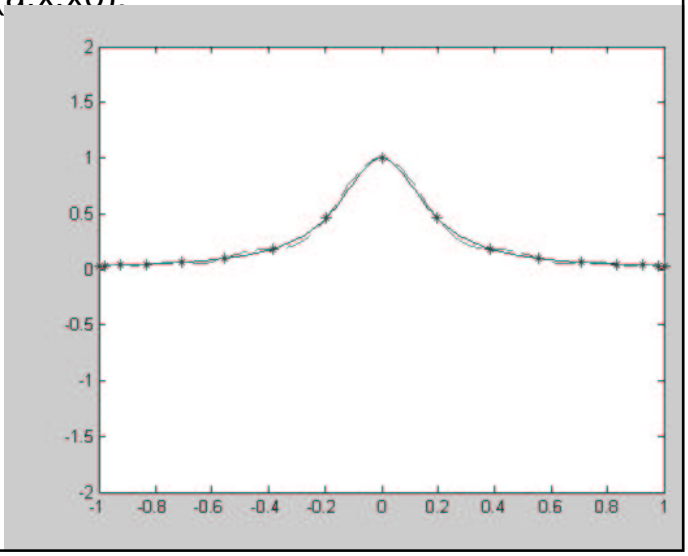


```
% least squares fit: degre 8 fit based on 17 pts.
[P,S] = polyfit(x,y,8);
pVal = polyval(P,x0);
```

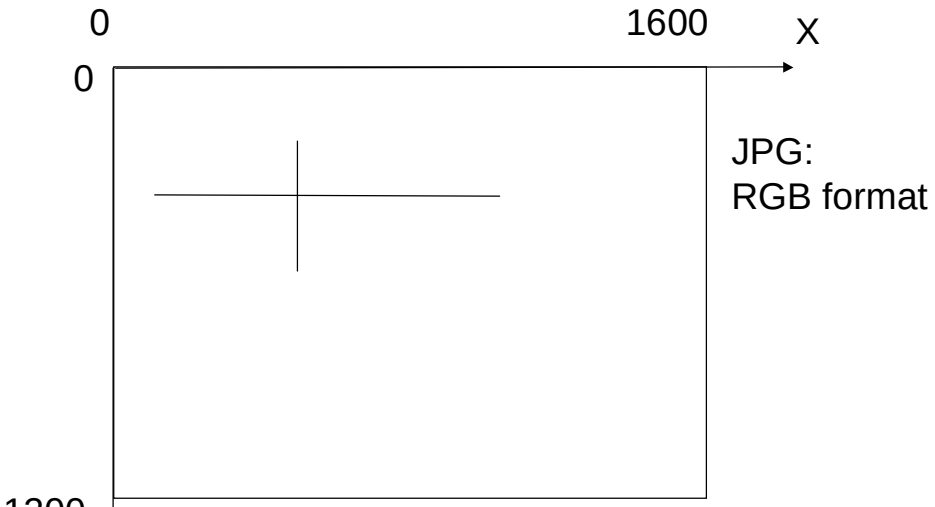


9/26/02

```
for i = 1:m+1                                   % Chebyshev Nodes
   x(i,1) = .5*(b+a) +.5*(b-a)*cos(pi*(i-1)/m) ; end
y =1./(1+ll*x.^2); a = InterpN(x,y);
pVal = HornerN(a,x,x0);
```



9/26/02



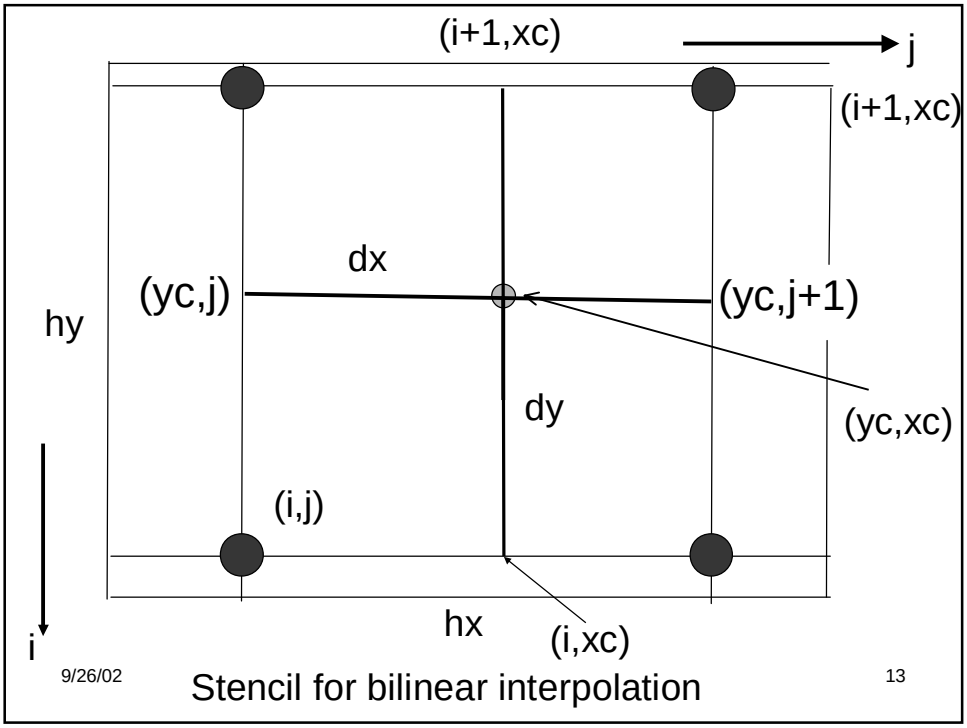

```
>[x,y]=ginput(1)
>[x,y]
     500    353
```

```
Size(X): 1200,1600, 3
class(X):   uint8
```

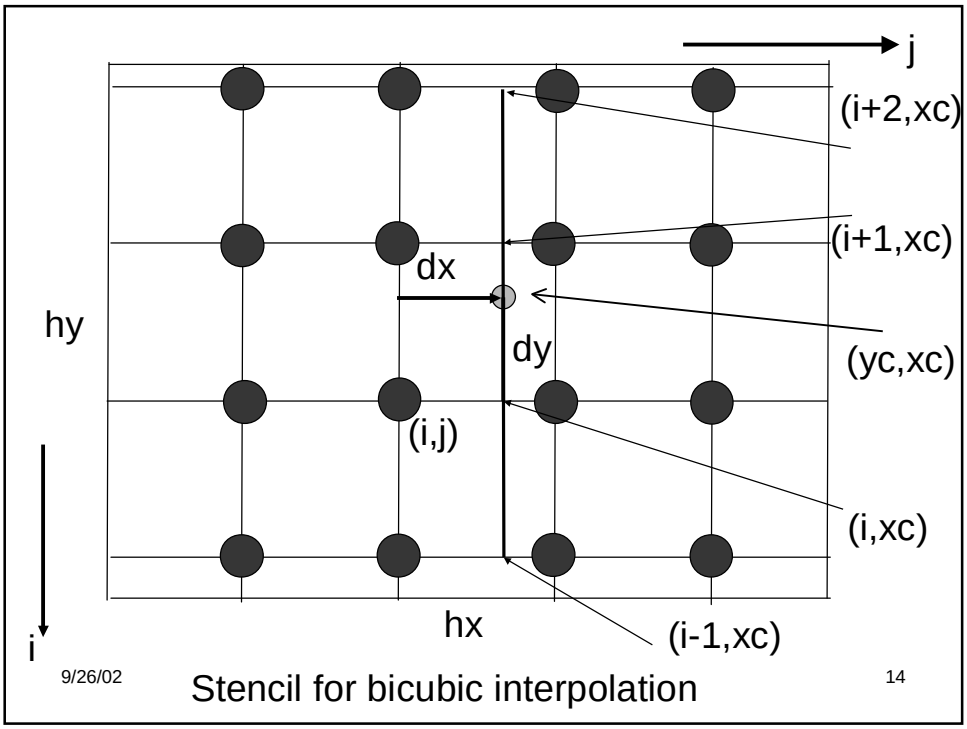
9/26/02 11

```
function z = LinInterp2Dc(xc,yc,a,b,c,d,fA)
[n,m] = size(fA(:,:,1));
hx = (b-a)/(n-1);
i  = max([1 ceil((xc-a)/hx)]);
dx = (xc - (a+(i-1)*hx))/hx;
hy = (d-c)/(m-1);
j  = max([1 ceil((yc-c)/hy)]);
dy = (yc - (c+(j-1)*hy))/hy;
for k=1:3
   z(k) = uint8((1-dy)*((1-
dx)*double(fA(i,j+1,k))+dx*double(fA(i+1,j+1,k))));
end
```

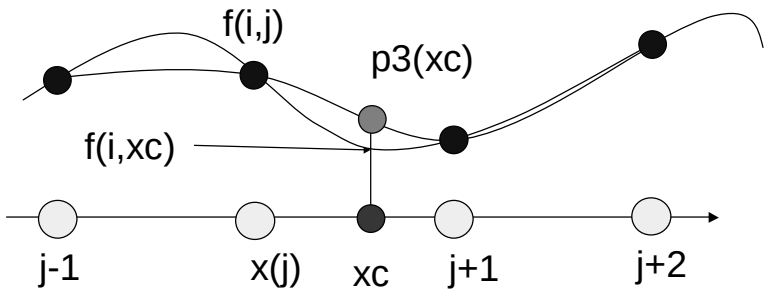
9/26/02 12



9/26/02 Stencil for bilinear interpolation 13



9/26/02 Stencil for bicubic interpolation 14




```
for ii=i-1:i+2
    [Px,S] = polyfit(x(j-1:j+2),f(ii,j-1:j+2),3);
    pc(ii,xc) = polyval(Px,xc);
end
[Py,S] = polyfit(y(i-1:i+2),pc(j-1:j+2,xc),3);
pc(yc,xc) = polyval(Py,yc);
```


Input-output structures

- x=input(‘string’)
- title=input(‘Title for plot:’,’string’)
- [x,y]=ginput(n)
- disp(‘a 3x3 magic square’)
- disp(magic(3))
- fprintf(‘%5.2f \n%5.2f\n’,pi^2, exp(1))

Writing to and reading from a file

```
fid = fopen('output','w')
  a=linspace(1,100,100);
  fprintf(fid,'%g degC = %g
  degF\n',a,32+1.8*a);
fid=fopen('output','r')
    x=fscanf(fid,'%g degC = %g degF');
    x=reshape(x,(length(x)/2),2)
```

9/26/02 17

```
----------------------------------------
I.
A=[1 2 3;4 5 6;7 8 9];
y=x'; %transpose
x=zeros(1,n);
length(x);
x=linspace(a,b,n);
a=x(5:-1:2);
%SineTable  (Help SineTable)
disp(sprintf('%2.0f  %3.0f ',k, a(k)));

----------------------------------------
```

```
---------------------------------------------------------------
II.
VECTOR OPERATIONS:

(1) vector:  scale, add, subtract (scalar*vector)
    2*[10 20 30] = [20 40 60]
(2) POINTWISE vector: multiply, divide, exponentiate (pointwise vector * vector)
    [2 3 4].*[10 20 30] = [20 60 120]
(for pointwise operations, dimensions must be identical!)
(3) Setting ranges for axes,
    Superimposing plots:
plot(x,y,x,exp(x),'o')
  axis([-40 0 0 0.1])
     using "hold on/off"
     extended parameter lists
(4) Subplots: subplot(m,n,k)
(5) Vector linear combos: matrix*vector
(6) Tensor Products (x is array, A is array of arrays of same shape as x)
       A=[f1(x) f2(x) ... fn(x)]
    A(k,:) k-th row; A(:,i) i-th column
(7) loops: loop until ~condition or for fixed no. of times
    while any(abs(term) > eps*abs(y))
    for term = 1:n
    end
(8) size(A) gives array dimensions (array, can be used to index other arrays)
    length(x) gives vector dimension
---------------------------------------------------------------
```

```
-----------------------------------------------------------------
II' (Miscellaneous topics)
rem(x,n): remainder
  if x == 1
(< <= == >= > ~=)
(& and; | or; ~ not; xor exclusive or)
  else
  endif
s3 = [s1 s2]; concatenation of strings, treated like arrays
[xmax, imax] = max(x); (max val, index)
rat = max(x)/x(1)
x(1) = input('Enter initial positive integer:')
density = sum(x<=x(1))/x(1);
```

Summary

- Interpolation
- vanderMonde, Newton, Least Squares
- 2d interpolatio": linear and cubic
- Color figures and uint8
- Smoothing of images

References

- find m-files at

  aix: ~coutsias/375
- C.F.van Loan,

  Intro to Sci.Comp. (2-3)