Numerical Marine Hydrodynamics

- Interpolation
  - Lagrange interpolation
  - Triangular families
  - Newton's iteration method
  - Equidistant Interpolation
  - Spline Interpolation
- Numerical Differentiation
- Numerical Integration
  - Error of numerical integration

Numerical Interpolation

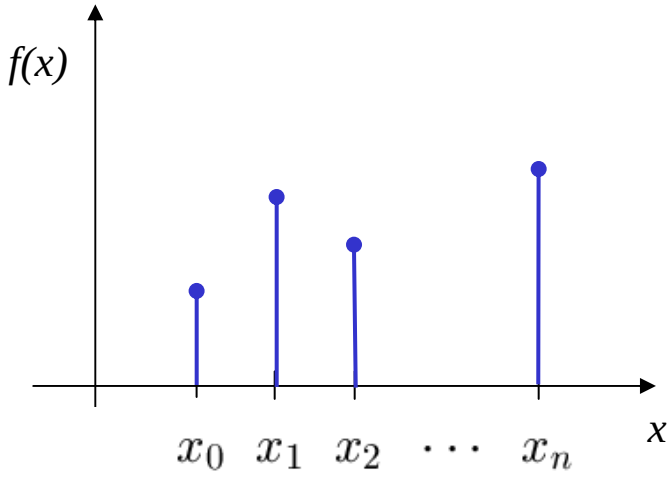
Given: $f(x_0) = f_0\ ,\ f(x_1) = f_1\ , \ldots f(x_n) = f_n$

Find $f(x)$ for $x \in [x_0, x_n]$

Purpose of numerical Interpolation

1. Compute intermediate values of a sampled function
2. Numerical differentiation – foundation for Finite Difference and Finite Element methods
3. Numerical Integration

Numerical Interpolation
Polynomial Interpolation



Interpolation

$$f(x) \simeq F(x)$$
$$f(x_i) = F(x_i)$$

$F(x)$ Interpolation function

Polynomial Interpolation

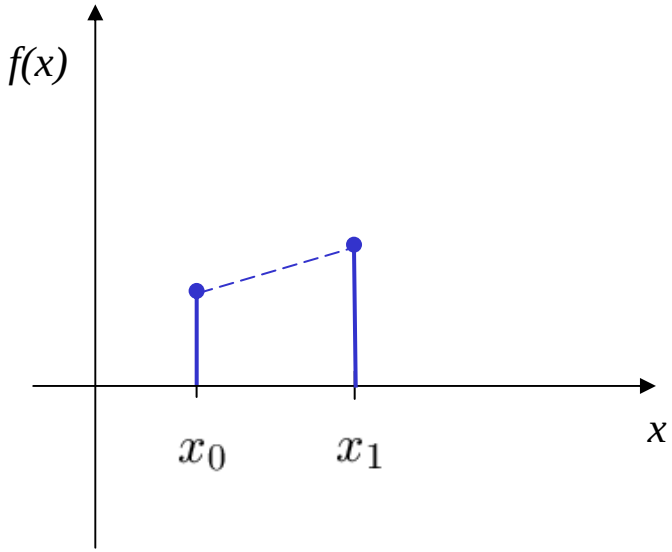
$$F(x) = p(x) = a_0 x^n + a_1 x^{n-1} \cdots a_{n-1} x + a_n$$

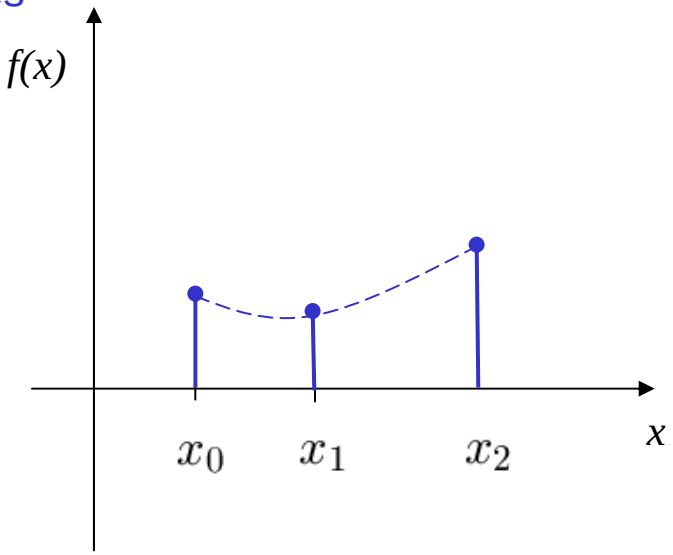
Coefficients: Linear System of Equations

$$\begin{aligned} f_0 &= a_0 x_0^n + a_1 x_0^{n-1} \cdots a_{n-1} x_0 + a_n \\ f_1 &= a_0 x_1^n + a_1 x_1^{n-1} \cdots a_{n-1} x_1 + a_n \\ &\vdots \\ f_n &= a_0 x_n^n + a_1 x_n^{n-1} \cdots a_{n-1} x_n + a_n \end{aligned}$$

Numerical Interpolation
Polynomial Interpolation

Examples





Linear Interpolation

$$p(x) = f_0 + (f_1 - f_0)\frac{x - x_0}{x_1 - x_0}$$

Quadratic Interpolation

$$p(x) = a_0 x^2 + a_1 x + a_2$$

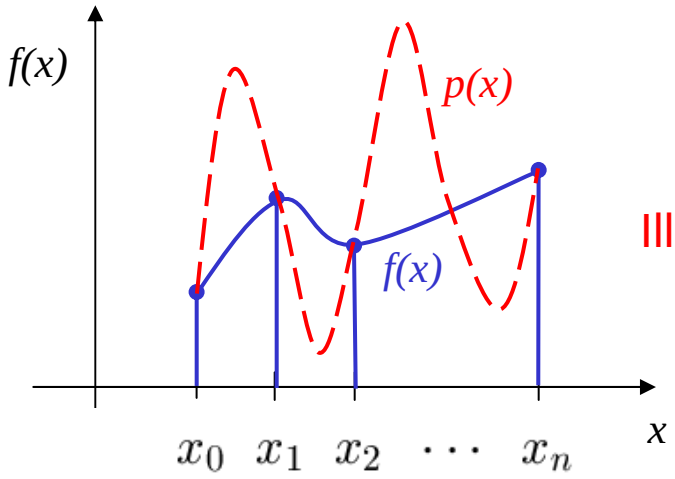
Numerical Interpolation
Polynomial Interpolation

Taylor Series

$$f(x) = p(x) + r(x) = f(x_0) + \sum_{i=1}^{n} \frac{f^{(i)}(x_0)}{(i+1)!}(x - x_0)^i + \frac{f^{(n+1)}(\xi)}{(n+1)!}(x - x_0)^{n+1}$$

Remainder

$$f(x) - p(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!}(x - x_0)^{n+1}$$



Requirement

$$f^{(n+1)}(\xi) \ll 1$$

Ill-conditioned for large n

Polynomial is unique, but how do we calculate the coefficients?

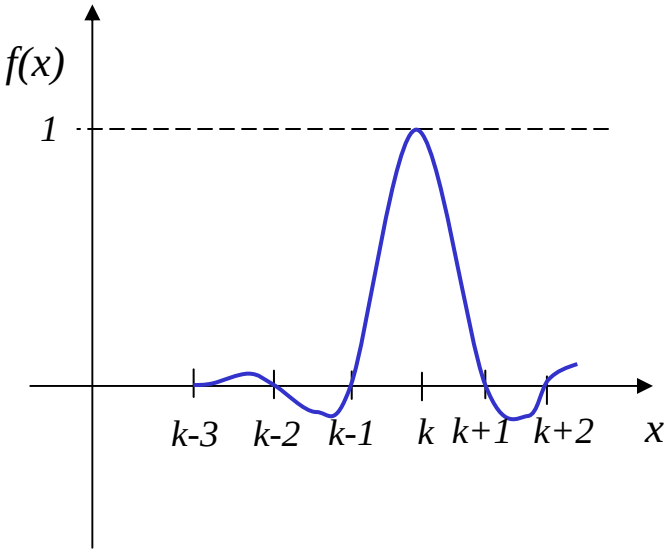
Numerical Interpolation
Lagrange Polynomials

$$p(x) = \sum_{k=0}^{n} L_k(x) f(x_k) = \sum_{k=0}^{n} L_k(x) f_k$$

$$L_k(x) = \sum_{i=0}^{n} \ell_{ik} x^i$$

$$L_k(x_i) = \delta_{ki} = \begin{cases} 0 & k \neq i \\ 1 & k = i \end{cases}$$

$$L_k(x) = \prod_{j=0, j\neq k}^{n} \frac{x - x_j}{x_k - x_j}$$



Difficult to program
Difficult to estimate errors
Divisions are expensive

Important for numerical integration

Numerical Interpolation
Triangular Families of Polynomials

Ordered Polynimials

$$p(x) = c_0\phi_0(x) + c_1\phi_1(x) + \cdots + c_n\phi_n(x)$$

where

$$\begin{aligned}
\phi_0(x) &= a_{00} \\
\phi_1(x) &= a_{10} + a_{11}x \\
\phi_1(x) &= a_{20} + a_{21}x + a_{22}x^2 \\
&\cdot \\
&\cdot \\
\phi_n(x) &= a_{n0} + a_{n1}x + \cdots\cdots\cdots + a_{nn}x^n
\end{aligned}$$

Special form convenient for interpolation

$$\begin{aligned}
\phi_0(x) &= 1 \\
\phi_1(x) &= x - x_0 \\
\phi_2(x) &= (x - x_0)(x - x_1) \\
&\cdot \\
&\cdot \\
\phi_n(x) &= (x - x_0)(x - x_1)\cdots(x - x_{n-1})
\end{aligned}$$

Coefficients

$$\begin{aligned}
f(x_0) = p(x_0) &= c_0 \\
f(x_1) = p(x_1) &= c_0 + c_1(x_1 - x_0) \\
f(x_2) = p(x_2) &= c_0 + c_1(x_2 - x_0) + c_2(x_2 - x_0)
\end{aligned}$$

$c_0,\ c_1,\ \ldots c_n$ found by recursion

.

Numerical Interpolation
Triangular Families of Polynomials

Polynomial Evaluation
Horner's Scheme

$$\begin{aligned} f(x) &\simeq c_0\phi_0(x) + c_1\phi_1(x)\cdots \\ &= c_0 + (x-x_0)(c_1 + (x-x_1)(c_2 + (x-x_2)(\cdots)))) \end{aligned}$$

Remainder – Interpolation Error

$$r(x) = f(x) - p(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!}(x-x_0)\cdots(x-x_n)$$

Numerical Interpolation
Newton's Iteration Formula

Standard triangular family of polynomials

$$
\begin{aligned}
f(x) &= p(x) + r(x) \\
&= c_0 + c_1(x - x_0) \cdots + c_n(x - x_0) \cdots (x - x_{n-1}) \\
&\quad + \frac{f^{(n+1)}(\xi)}{(n+1)!}(x - x_0) \cdots (x - x_n)
\end{aligned}
$$

Divided Differences

$$f(x_0) = c_0 \Rightarrow c_0 = \boxed{f(x_0)}$$

$$f(x_1) = c_0 + c_1(x_1 - x_0) \Rightarrow c_1 = \frac{f(x_1) - f(x_0)}{x_1 - x_0} = \boxed{f[x_0, x_1]}$$

$$f(x_2) = c_0 + c_1(x_2 - x_0) + c_2(x_2 - x_0)(x_2 - x_1)$$

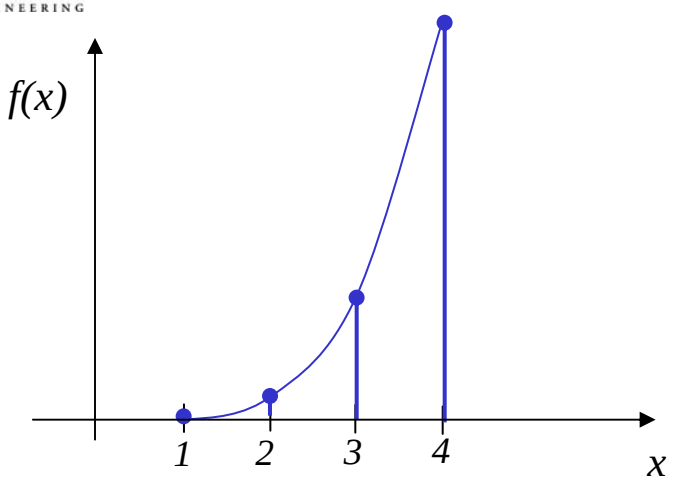
$$c_2 = \frac{\frac{f(x_2) - f(x_1)}{x_2 - x_1} - \frac{f(x_1) - f(x_0)}{x_1 - x_0}}{x_2 - x_0} = \frac{f[x_1, x_2] - f[x_0, x_1]}{x_2 - x_0} = \boxed{f[x_0, x_1, x_2]}$$

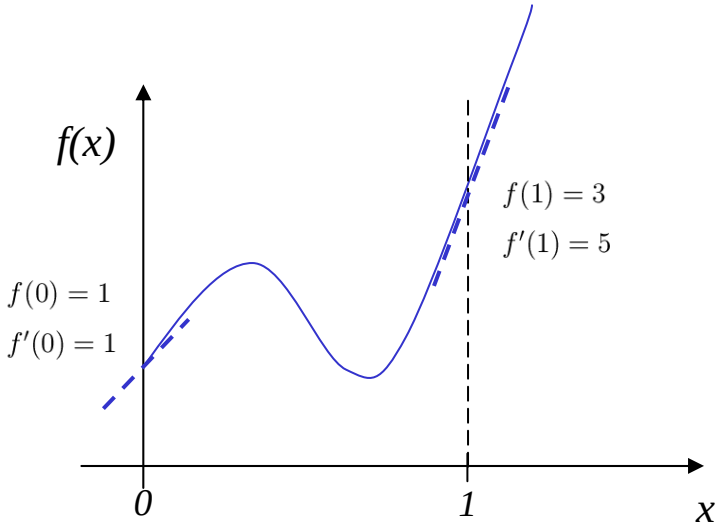
$$c_n = \boxed{f[x_0, x_1, \ldots, x_n]} = \frac{f[x_1, \ldots, x_n] - f[x_0, \ldots x_{n-1}]}{x_n - x_0}$$

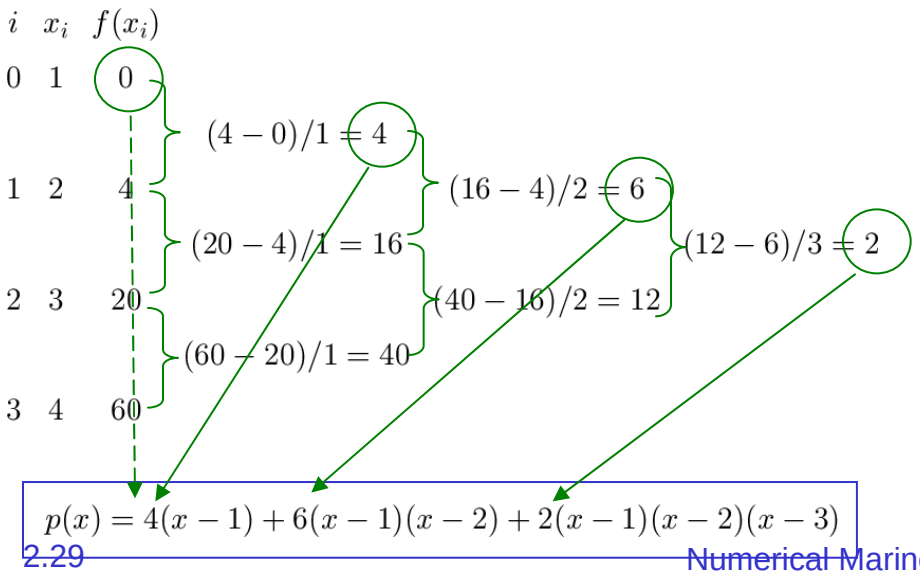
Newton's Computational Scheme

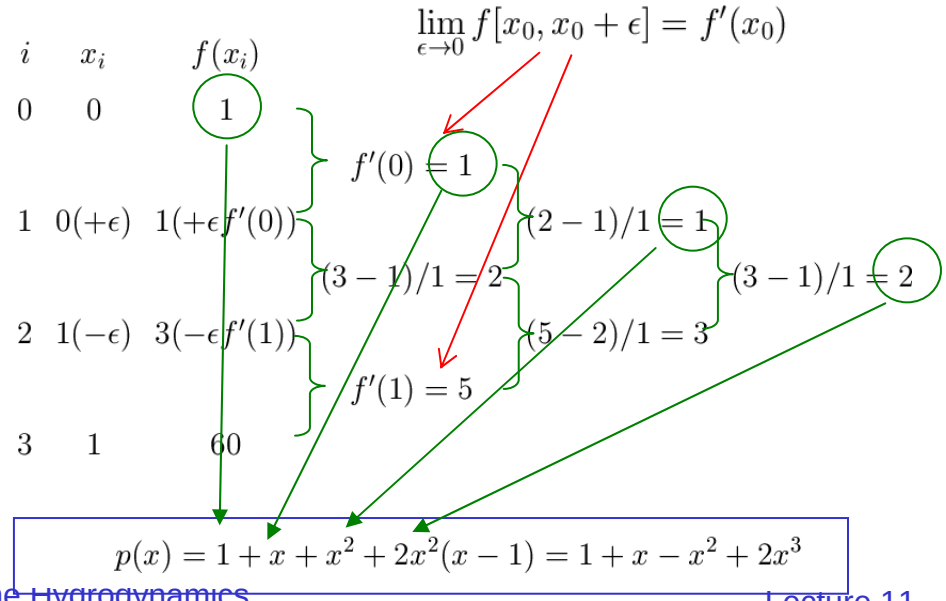
		c_0	c_1	c_2	c_3
x_0	$f(x_0)$				
			$f[x_0, x_1]$		
x_1	$f(x_1)$			$f[x_0, x_1, x_2]$	
			$f[x_1, x_2]$		$f[x_0, x_1, x_2, x_3]$
x_2	$f(x_2)$			$f[x_1, x_2, x_3]$	
			$f[x_2, x_3]$		
x_3	$f(x_3)$				

Numerical Interpolation
Newton's Iteration Formula









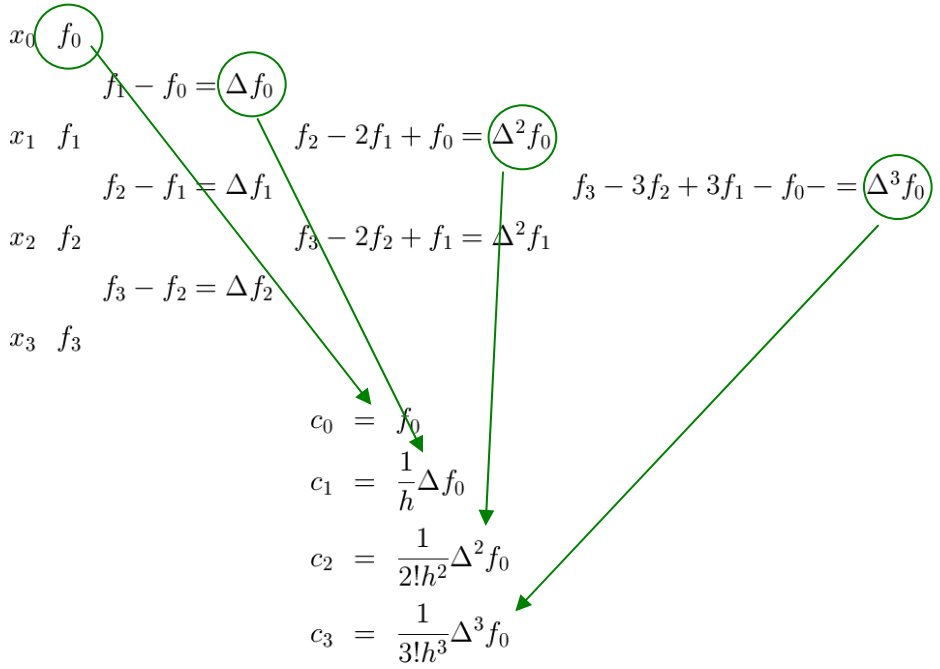
Numerical Interpolation
Equidistant Newton Interpolation

Equidistant Sampling

$$x_i = x_0 + ih$$

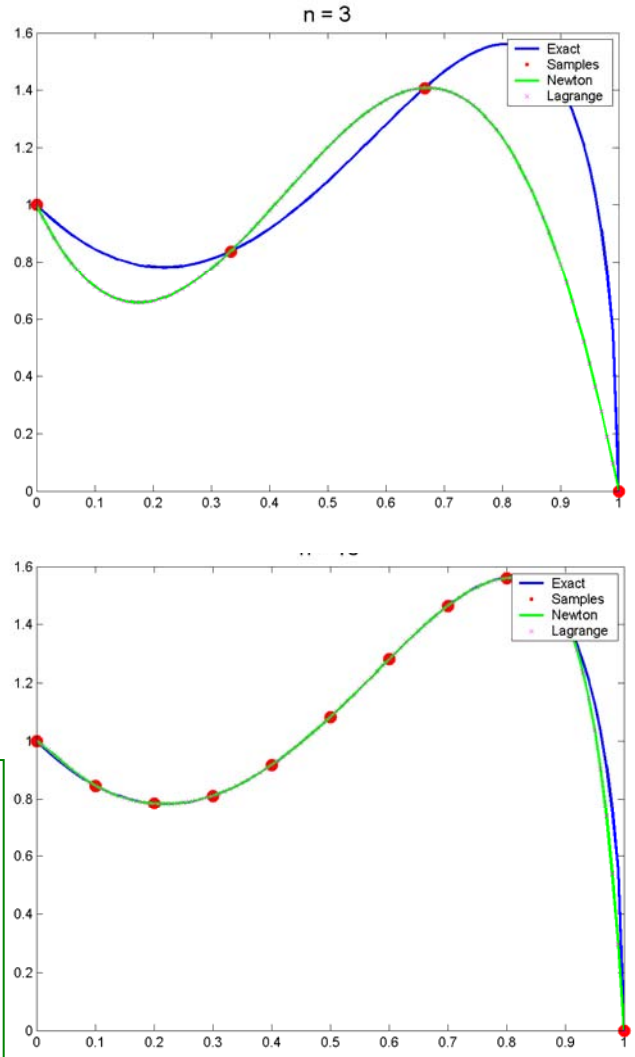
$$\begin{aligned}
f[x_0, x_1] &= \frac{f(x_1) - f(x_0)}{x_1 - x_0} = \frac{1}{h}(f_1 - f_0) = \frac{1}{h}\,\Delta f_0 \\
f[x_0, x_1, x_2] &= \frac{f[x_1, x_2] - f[x_0, x_1]}{x_2 - x_0} \\
&= \frac{1}{1 \cdot 2 \cdot h^2}(f_2 - 2f_1 + f_0) = \frac{1}{2!h^2}\Delta^2 f_0 \\
f[x_0, x_1, x_2, x_3] &= \frac{1}{3! \cdot h^3}(f_3 - 3f_2 + 3f_1 - f_0) = \frac{1}{3!h^3}\Delta^3 f_0
\end{aligned}$$

Divided Differences
Stepsize Implied



Numerical Interpolation
Newton's Iteration Formula

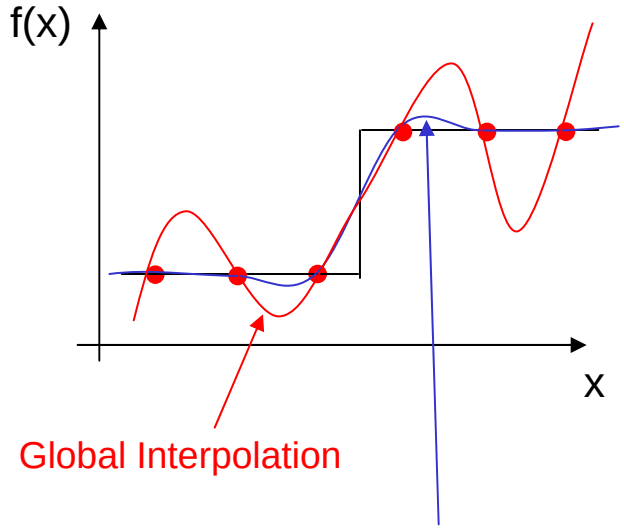
```
function[a] = interp_test(n)
%n=2
h=1/n
xi=[0:h:1]
f=sqrt(1-xi.*xi) .* (1 - 2*xi +5*(xi.*xi));
%f=1-2*xi+5*(xi.*xi)-4*(xi.*xi.*xi);

c=newton_coef(h,f)
m=101
x=[0:1/(m-1):1];
fx=sqrt(1-x.*x) .* (1 - 2*x +5*(x.*x));
%fx=1-2*x+5*(x.*x)-4*(x.*x.*x);

y=newton(x,xi,c);
hold off; b=plot(x,fx,'b'); set(b,'LineWidth',2);
hold on; b=plot(xi,f,'.r') ; set(b,'MarkerSize',30);
b=plot(x,y,'g'); set(b,'LineWidth',2);
yl=lagrange(x,xi,f);
b=plot(x,yl,'xm'); set(b,'Markersize',5);
b=legend('Exact','Samples','Newton','Lagrange')
b=title(['n = ' num2str(n)]); set(b,'FontSize',16);
```




```
function[y] = newton(x,xi,c)
% Computes Newton polynomial
% with coefficients c
n=length(c)-1
m=length(x)
y=c(n+1)*ones(1,m);
for i=n-1:-1:0
    cc=c(i+1);
    xx=xi(i+1);
    y=cc+y.*(x-xx);
end
```

```
function[c] = newton_coef(h,f)
% Computes Newton Coefficients
% for equidistant sampling h
n=length(f)-1
c=f; c_old=f; fac=1;
for i=1:n
fac=i*h;
for j=i:n
 c(j+1)=(c_old(j+1)-c_old(j))/fac;
end
c_old=c;
end
```

Numerical Interpolation
Splines



Linear Splines

$$
\begin{aligned}
f(x) &= f(x_0) + m_0(x - x_0)\ , \quad x_0 \le x \le x_1 \\
f(x) &= f(x_1) + m_1(x - x_1)\ , \quad x_1 \le x \le x_2 \\
&\ \ \vdots \\
f(x) &= f(x_{n-1}) + m_{n-1}(x - x_{n-1})\ , \quad x_{n-1} \le x \le x_n
\end{aligned}
$$

$$
m_i = \frac{f(x_{i+1}) - f(x_i)}{x_{i+1}) - x_i}
$$

Numerical Interpolation
Splines

Quadratic Splines

$$f_i(x) = a_i x^2 + b_i x + c_i \ , \quad x_{i-1} \le x \le x_i$$

3n coefficients

Continuity

$$\begin{aligned} a_{i-1}x_{i-1}^2 + b_{i-1}x_{i-1} + c_{i-1} &= f(x_{i-1}) \\ a_i x_{i-1}^2 + b_i x_{i-1} + c_i &= f(x_{i-1}) \end{aligned}$$

2(n-1) conditions

End Conditions

$$\begin{aligned} a_1 x_0^2 + b_1 x_0 + c_1 &= f(x_0) \\ a_n x_n^2 + b_n x_n + c_n &= f(x_n) \end{aligned}$$

2 conditions – 2n total

Derivatives

$$f_i'(x) = 2a_i x + b_i$$

$$2a_{i-1}x_{i-1} + b_{i-1} = 2a_i x_{i-1} + b_i$$

n-1 conditions – 3n-1 total

$$a_1 = 0$$

1 condition – 3n total

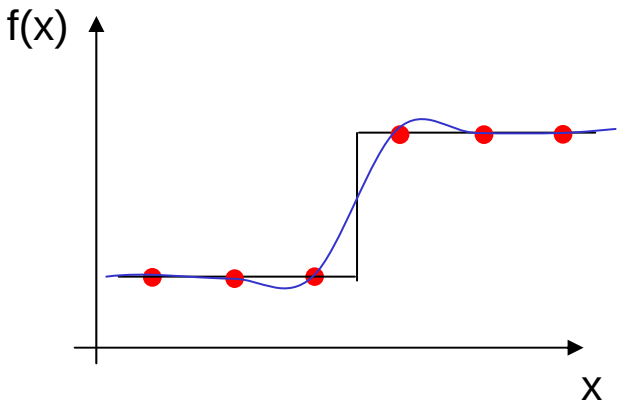
Non-Unique!

Numerical Interpolation
Splines

Quadratic Splines

$$f_i(x) = a_i x^3 + b_i x^2 + c_i x + d_i \ , \quad x_{i-1} \le x \le x_i$$

4n unknowns



Continuity Conditions

1. Continuity – 2n-2 conditions
2. End conditions – 2 conditions
3. Continuous 1st derivatives – n-1 conditions
4. Continuous 2nd derivatives – n-1 conditions
5. Second derivatives at end points – 2 conditions