# Numerical Methods (31/Nov — 15/Nov )

Optim, in 1-d

$f : \mathbb{R} \to \mathbb{R}$

Newton's Method

$\min f(x) \longrightarrow$ Find $x$ such that $f'(x) = 0$

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

Unconstrained Minimization in n-Dimension

$f : \mathbb{R}^n \to \mathbb{R}$

General Framework

$$x_{k+1} = x_k + \alpha_k p_k$$

$x_k \in \mathbb{R}^n$, $\alpha_k \in \mathbb{R}$

$p_k$ is a descend direction or direction vector

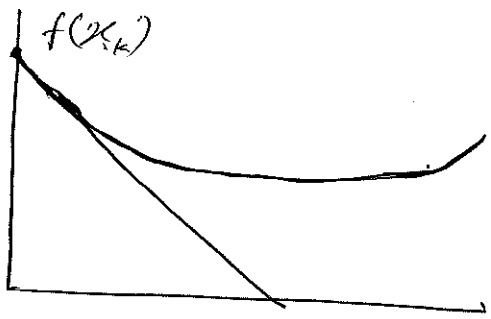
1) Gradient Method

[Steepest descent method]

$$p_k = -\nabla f(x_k)$$

① Armijo (Goldstein) Line Search

$$\varphi(\alpha) = f(x_k + \alpha p_k)$$




Given $x, p \in \mathbb{R}^n$ and $\alpha_{init} = 1$, $\tau = \frac{1}{2}$, $\beta = \frac{1}{L}$

$\alpha = \alpha_{init}$

while $f(x+\alpha p) \geq f(x) + \alpha \cdot \beta \cdot \nabla f(x)^T p$

$\alpha = \tau \alpha$

end

$$\varphi(\alpha) = \nabla f(x_k + \alpha p_k)^T p_k$$

$$\varphi'(0) = \nabla f(x_k)^T p_k$$

Test Problem:

$$f : \mathbb{R}^2 \longrightarrow \mathbb{R}$$

$$f(x,y) = (1-x^2) + 10(y-x^2)^2 \qquad \text{[Rosenbrock Function]}$$

2) Newton's Method

for unconstrained minimization in n-dim.

$[f : \mathbb{R}^n \to \mathbb{R}]$

$$f(x+p) \approx \underbrace{f(x) + \nabla f(x)^T p + \frac{1}{2} p^T H_f(x) p}_{\bar{f}(p)}$$

$$\nabla \bar{f}(p) = \nabla f(x) + H_f(x) \cdot p$$

$$= 0 \quad \text{for} \quad H_f(x) \cdot p = -\nabla f(x) \qquad (\star)$$

$\Rightarrow$ $x$ is $x_k$, $p$ is $p_k$

We solve $(\star)$ for $p_k$

$x_{k+1} = x_k + p_k$

3) BFGS Method

Solve

$$B_k p_k = -\nabla f(x_k)$$

Start with $B = I$

At each iteration:

$$B_{k+1} = B_k + \frac{y_k y_k^T}{y_k^T p_k} - \frac{(B_k p_k)(B_k p_k)^T}{p_k^T B_k p_k}$$

$$y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$$

we again use the line searching method to get $\alpha_k$

$$x_{k+1} = x_k + \alpha_k p_k$$

4) Nonlinear Conjugate Gradient Method

$$\boxed{P_{k+1} = -g_{k+1} + \beta_{k+1} P_k} \qquad g_{k+1} = \nabla f(x_{k+1})$$

$$\beta_{k+1} = \frac{g_{k+1}^T g_{k+1}}{g_k^T g_k}$$

8-Nov

## Non-Linear Least Squares

Let $r: \mathbb{R}^n \longrightarrow \mathbb{R}^m$, $m > n$

Would like : Find $x \in \mathbb{R}^n$ such that $r(x) = 0$

Instead look for $x$ such that,

$$\phi(x) = \frac{1}{2}\left[ r_1(x)^2 + r_2(x)^2 + \ldots + r_m(x)^2 \right] = \frac{1}{2} r(x)^T r(x)$$

$$= \frac{1}{2} \| r(x) \|_2^2$$

is minimized.

Jacobian of $r$: $\left[ J_r(x) \right]_{ij} = \dfrac{\partial r_i(x)}{\partial x_j}$

$$\left[ \nabla \phi(x) \right]_k = \frac{\partial \phi}{\partial x_k} = r_1(x) \frac{\partial r_1}{\partial x_k} + r_2(x) \frac{\partial r_2}{\partial x_k}$$

$$+ \ldots + r_m(x) \frac{\partial r_m}{\partial x_k}$$

$$= r_1(x) \left[ J_r(x) \right]_{1k} + \ldots + r_m(x) \left[ J_r(x) \right]_{mk}$$

$$= \left[ J_r(x)^T \cdot r(x) \right]_k$$

$$\underset{n\times 1}{\nabla \phi(x)} = \underset{n\times m}{J_r(x)^T} \cdot \underset{m\times 1}{r(x)}$$ (Check on specification)

Entries of the Hessian Matrix:

$$\left[H_\phi(x)\right]_{jk} = \frac{\partial^2 \phi}{\partial x_j \partial x_k} = \frac{\partial r_1}{\partial x_j}\cdot\frac{\partial r_1}{\partial x_k} + r_1 \cdot \frac{\partial^2 r_1}{\partial x_j \partial x_k}$$

$$+ \cdots + \frac{\partial r_m}{\partial x_j}\frac{\partial r_m}{\partial x_j} + r_m \frac{\partial^2 r_m}{\partial x_j \partial x_k}$$

$$= \sum_{i=1}^{m} \frac{\partial r_i}{\partial x_j}\cdot\frac{\partial r_i}{\partial x_k}$$

$$+ \sum_{i=1}^{m} r_i \left[H_{r_i}(x)\right]_{jk}$$

$$H_\phi(x) = J_r(x)^T J_r(x) + \sum_{i=1}^{m} r_i \cdot \left[H_{r_i}(x)\right]$$

## Newton Iteration for Minimization

1) Form: $H_\phi(x_k)$, $\nabla \phi(x_k)$

2) Solve: $[H_\phi]\cdot p = -\nabla\phi \rightarrow p_k$ (solution)

3) Update: $x_{k+1} = x_k + p_k$

# Gauss-Newton Method for Non-Linear Least Squares

- Replace $H_\phi(x)$ by $J_r^T J_r$
- Solve $(J_r(x)^T J_r(x))p = -\nabla\phi(x)$ (A) $\longrightarrow p_k$ (solve)
- Update : $x_{k+1} = x_k + p_k$

(A) $\Rightarrow$ $J_r^T J_r \cdot p = -J_r^T$

Note that $J^T J p = -J^T r$

are the normal equations for the linear least squares problem

$$J_r(x) \cdot p \approx -r$$

Example

$y = Ce^{kt}$ on $(t_1, y_1) \dots (t_m, y_m)$

$r_i = y_i - x_1 e^{x_2 t_i}$

$r : \mathbb{R}^2 \longrightarrow \mathbb{R}^m$

$$\frac{\partial r_i}{\partial x_1} = -e^{x_2 t} \qquad \frac{\partial r_i}{\partial x_2} = -x_i t e^{x_2 t}$$

$$J_r(x) = \begin{bmatrix} \vdots \\ \vdots \end{bmatrix}, \quad r(x) = \begin{bmatrix} \vdots \\ \vdots \end{bmatrix}$$

Book: Section on Non-Linear Least Squares.

L-M method to solve ill-conditioning

$$(J^T J + \alpha I)p = -J^T r$$

∴ Regularization

Linear Programming

In general: $\min f(x)$ such that $g(x) = 0$, $h(x) \leq 0$

↳ $f, g, h$ are all linear functions.

$$f(x) = c^T x \qquad \min \; c^T x$$

$$g(x) = 0 \Rightarrow Ax = b \qquad \text{such that}$$

$$h(x) \leq 0 \Rightarrow A_2 x \leq b_2 \qquad lb \leq x \leq ub$$

$$0 \leq x \leq M$$

To solve a linear programming problem we use linprog

$$x = \text{linprog}(f, A, b, A_2, b_2, lb, ub)$$

(A, b: ineq; $A_2$, $b_2$: eq)

e.g $f = \begin{bmatrix} -8 \\ -11 \end{bmatrix}$

$A = \begin{bmatrix} 5 & 4 \\ -1 & 3 \end{bmatrix}$ $b = \begin{bmatrix} 40 \\ 12 \end{bmatrix}$

$A2 = [\ ]$ $b_2 = [\ ]$ (If no equality constraint)

$lb = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ $ub = [\ ]$ (Example)

# Quadratic Programming Problem

$$f(x) = \frac{1}{2} x^T C x + x^T d$$

$$g(x) = Ex - f = 0$$

Important for Finals

$$\boxed{\min f(x) \text{ s.t. } g(x) = 0}$$

$$\mathcal{L}(x, \lambda) = f(x) + \lambda^T g(x)$$

$$= \frac{1}{2} x^T C x + x^T d + \lambda^T (Ex - f)$$

$$\nabla \mathcal{L}(x, \lambda) = \begin{pmatrix} \nabla_x \mathcal{L} \\ \nabla_\lambda \mathcal{L} \end{pmatrix} = \begin{pmatrix} Cx + d + E^T \lambda \\ Ex - f \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

$$\Rightarrow \begin{bmatrix} C & E^T \\ E & 0 \end{bmatrix} \begin{bmatrix} x \\ \lambda \end{bmatrix} = \begin{bmatrix} -d \\ f \end{bmatrix}$$

Problem

$\min f(x)$

s.t. $g(x) = 0$

$(f : \mathbb{R}^n \to \mathbb{R},\ g : \mathbb{R}^n \to \mathbb{R}^m)$

$\nabla \mathcal{L} : \mathbb{R}^{n+m} \to \mathbb{R}^{n+m}$

$$\mathcal{L}(x, \lambda) = f(x) + \lambda^T g(x)$$

$$\nabla_x \mathcal{L}(x, \lambda) = \nabla f(x) + J_g(x)^T \lambda \stackrel{!}{=} 0_n \qquad \nabla_\lambda \mathcal{L}(x, \lambda) = g(x) \stackrel{!}{=} 0_m$$

$$\begin{pmatrix} x_{k+1} \\ \lambda_{k+1} \end{pmatrix} = \begin{pmatrix} x_k \\ \lambda_k \end{pmatrix} - \left( H_2(x_k, \lambda_k) \right)^{-1} \begin{pmatrix} \nabla_x \mathcal{L} \\ \nabla_\lambda \mathcal{L} \end{pmatrix}$$

(★) $$H_2(x_k, \lambda_k) = \begin{bmatrix} B_k & J_g(x_k)^T \\ J_g(x_k) & O \end{bmatrix} \begin{matrix} n \\ m \end{matrix} \quad \begin{matrix} n & m \end{matrix}$$

KKT Matrix for QP problem →

$$B_k = H_f(x) + \sum (\lambda_k)_i H_{g_i}(x_k) \in \mathbb{R}^{n \times n}$$

MATLAB: `H_L = [B, Jg'; Jg, Om]`

⊙ Solve a QP at each iteration

→ Sequential Quadratic Programming (SQP)

$$QP: \quad \tilde{f}(p) = \frac{1}{2} p^T B p + p^T \left[ \nabla f(x) + J_g(x) \lambda \right]$$

$$\tilde{g}(p) = g(x) + J_g(x) \cdot \lambda$$

∴ ⊙ Separately describe in words what the code is doing

## Inequality Constraints

$f(x) \stackrel{!}{=} \min$ such that $g(x)=0$, $h(x) \leq 0$

Active Set Strategy

A modification of (*) can be

$$\begin{bmatrix} A & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} b \\ c \end{bmatrix}$$

KKT / Saddle Point

## Interpolation

Given points $(t_i, y_i)$ $i = 1, 2, \ldots, m$

Find $f$ such that $f(t_i) = y_i$

($f$ interpolates the data)

## Polynomial Interpolation

(Other possibilities:
- piecewise polynomial
- rational functions
- trig functions )

Related Problem:

Given $f$, find $p \in P_n$ that approximates $f$

$$\mathcal{P}_n = \text{Span}\{1, x, \dots, x^n\}$$

$$p(k) = \sum_{j=0}^{n} c_j x^j = c_0 + c_1 t + c_2 t^2 + \dots + c_n t^n$$

$\{1, x, \dots, x^n\}$ is the monomial basis.

Let $\{\phi_1, \dots, \phi_{n+1}\}$ be a basis of $\mathcal{P}_n$.

$$p(t_i) = \sum_{j=1}^{n+1} c_j \underbrace{\phi_j(t_i)}_{A_{ij}} \Rightarrow p(t_i) = y_i$$

$$\Rightarrow A \cdot c = y$$

⊙ $A_n$ interpolation exists and is unique if $A$ is non-singular

Monomial basis:

$$A_{ij} = t_i^{j-1} \quad (\text{as } \phi_j = t^{j-1})$$

$$A = \begin{bmatrix} 1 & t_1 & t_1^2 & \dots & t_1^n \\ 1 & t_2 & t_2^2 & \dots & t_2^n \\ \vdots & & & & \\ 1 & t_{n+1} & t_{n+1}^2 & \dots & t_{n+1}^n \end{bmatrix}$$

(?) → Vandermonde Matrix : $V$

$$\det(V) = \prod_{i>j} (t_i - t_j)$$

$\neq 0$, provided $t_i$ distint

# Horner's Scheme

$$p(t) = c_1 + c_2 t + \dots + c_n t^{n-1}$$

$$= c_1 + t(c_2 + t(c_3 + \dots + (c_{n-1} + c_n t))\dots)$$

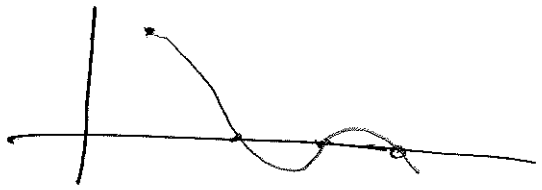
Code:

```
pt = c(n)
for  i = n-1:-1:1
     pt = c(i) + t · p(t)
end
```

# Lagrange Interpolation

Given $t_0, t_1, t_2, t_3$

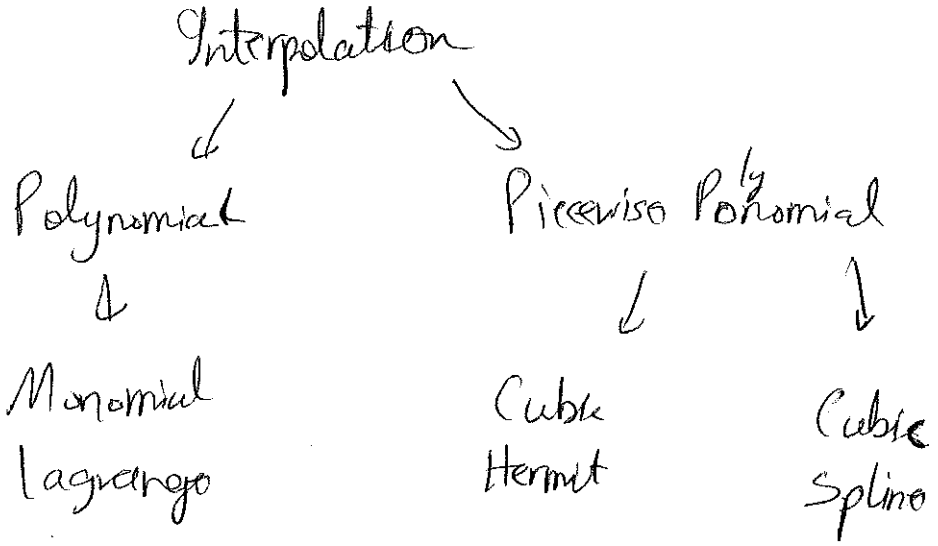
$L_1(t)$



$$L_i(t) = \frac{\prod_{\substack{k=1 \\ k \neq i}}^{n} (t - t_k)}{\prod_{\substack{k=1 \\ k \neq i}}^{n} (t_i - t_k)}$$

15/Nov

# Lagrange Interpolation




Lag Intr.

$$L_j(t) = \frac{\prod_{\substack{k=1 \\ k \neq j}}^{n} (t - t_k)}{\prod (t_j - t_k)}$$

$Ac = y$

$A = I \Longrightarrow (t_i, y_i)$

A lower triangular for Newton Method

# Newton Interpolation

Newton Basis $\{1, (t-t_1), (t-t_1)(t-t_2), \ldots, (t-t_1)(t-t_2)\ldots(t-t_{n-1})\}$

# Divided Differences

$(t_1, y_1), \ldots (t_k, y_k)$

$f[t_i] = y_i$

$f[t_1, \ldots, t_k] =$

$$\frac{f[t_2, \ldots, t_k] - f[t_1, \ldots, t_{k-1}]}{t_k - t_1}$$

$t_1 \; f[t_1] = y_1 \longrightarrow$

$f[t_2] = y_2 \longrightarrow$

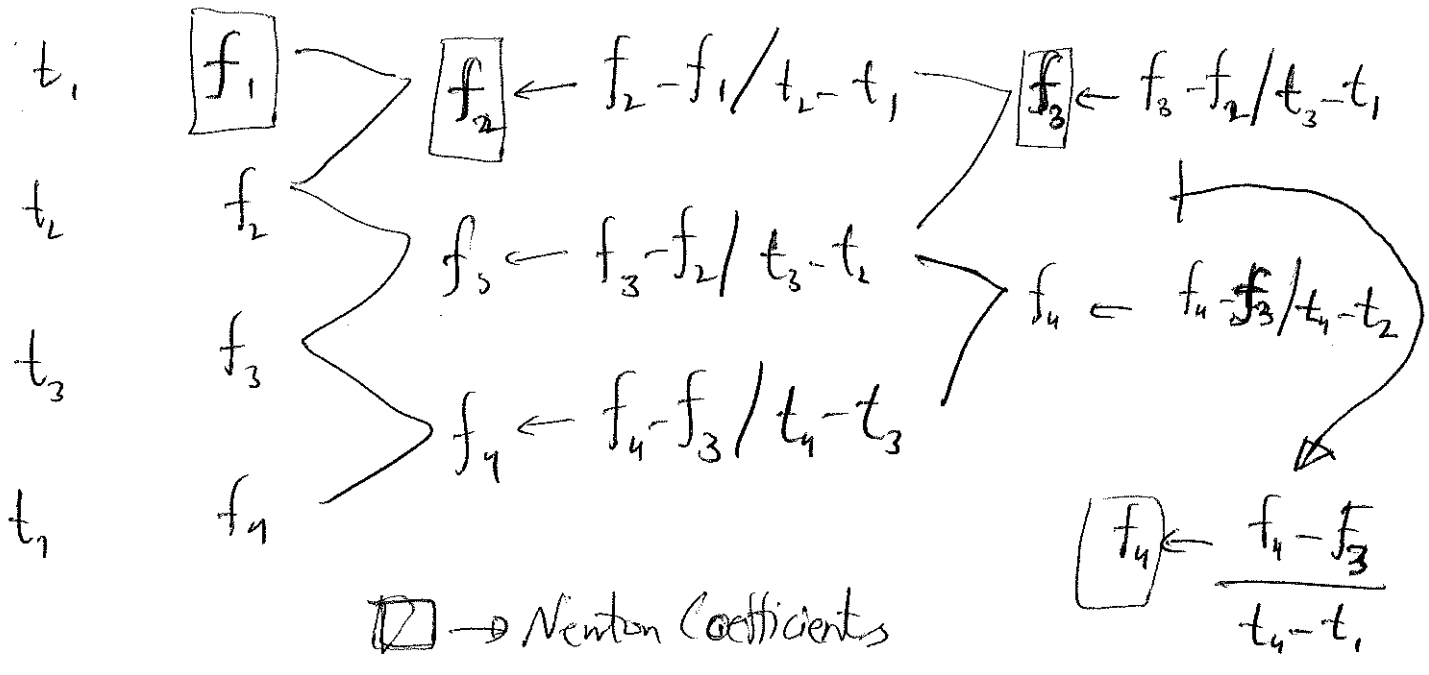
$$f[t_1, t_2] = \frac{y_2 - y_1}{t_2 - t_1}$$

$$f[t_1, t_2] = \frac{f[t_2] - f[t_1]}{t_2 - t_1} = \frac{y_2 - y_1}{t_2 - t_1}$$

$$f[t_2, t_3] = \frac{y_3 - y_2}{t_3 - t_2}$$

$$\Longrightarrow f[t_1, t_2, t_3] = \frac{f[t_2, t_3] - f[t_1, t_2]}{t_3 - t_1}$$

Similarly we get $f[t_1, t_2, t_3, t_4] = \dfrac{f[t_2, t_3, t_4] - f[t_1, t_2, t_3]}{t_4 - t_1}$



$t, f \in \mathbb{R}^n$

for $k = 1 : n-1$

  for $j = n : -1 : k+2$

$$f(j) = \frac{f(j) - f(j-1)}{t(j) - t(j-k)}$$

## Evaluating the Newton Polynomial

$$p(t) = c_1 + c_2(t-t_1) + c_3(t-t_1)(t-t_2) + c_4(t-t_1)(t-t_2)(t-t_3)$$

$$= c_1 + (t-t_1)\left[c_2 + c_3(t-t_2) + c_4(t-t_2)(t-t_3)\right]$$

$pt = c_4$ ;

$pt = c_3 + pt \cdot (t - t_3)$

$pt = c_2 + pt\,(t - t_2)$

$pt = c_1 + pt\,(t - t_1)$

```
pt = c(n)
for i = n-1:-1:1
    pt = c(i) + pt*(t - t(i))
end
```

Given the set

| $t$ | 0 | 1 | 2 |
|---|---|---|---|
| $y$ | 3 | 1 | 2 |

1) Find interpolating polynomial in Newton Form. Express in Monomial ~~Form~~ basis.

2) Find in Lagrange Form. Express in monomial basis.

(1) $p(t) = c_1 + c_2(t-t_1) + c_3(t-t_1)(t-t_2) \Rightarrow d_1 + d_2 x + d_3 x^2$

## Error in Polynomial Interpolation

$f : [a,b] \to \mathbb{R}$, $k+1$ times diffable, $P_k \in \mathcal{P}_k$ be the interpol polynomial at $t_1, t_2 \dots t_k$

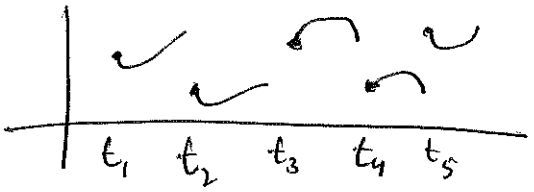
$$f(t) - P_k(t) = \frac{f^{(k+1)}(\xi)}{(k+1)!} \prod_{i=1}^{k+1} (t - t_i) \qquad a < \xi < b$$

If $|f^{(k+1)}(t)| \le M$, $\Rightarrow |e(t)| \le \frac{M}{(k+1)!} \prod_{i=1}^{k+1} (t - t_i) \quad \forall t \in [a,b]$

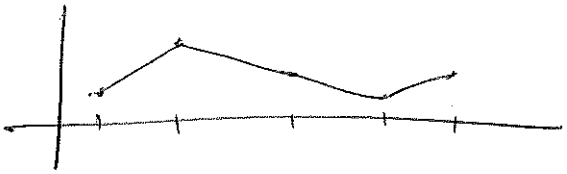
# Runge Phenomenon, Runge Function

$$f(x) = \frac{1}{1+25x^2}$$

## Piecewise Polynomial Interpolation



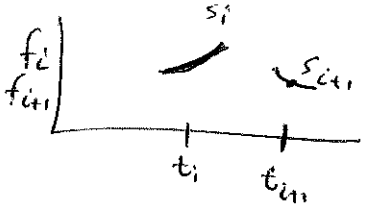

## Piecewise Linear



## Piecewise Cubic Interpolation

### 1) Hermite Interpol

Given $t_i, f_i, s_i$ (slope)




$$p_i(t) = c_{1,i} + c_{2,i}(t-t_i) + c_{3,i}(t-t_i)^2 + c_{4,i}(t-t_i)^3$$

$c_{1,i} = f_i$

$c_{2,i} = s_i$

$$c_{4,i} = \frac{s_{i+1} + s_i + 2f[t_i, t_{i+1}]}{(\Delta t_i)^2}$$

$$c_{3,i} = \frac{f[t_i, t_{i+1}] - s_i}{\Delta t_i} - c_{4,i}\Delta t_i$$

# Cubic Splines

Given $t_i, f_i$

find the interpolating piecewise cubic polynomial [spline]

$\Rightarrow$ Determine $s_i$ by enforcing the condition that spline be in $C^2$ (twice differentiable)

$$p_i'(t_{i+1}) = p_{i+1}'(t_{i+1})$$

$$p_i''(t_{i+1}) = p_{i+1}''(t_{i+1})$$

$\therefore$ these conditions are used for computing $s_i$