# **Newton-Raphson Algorithm**

# 

**Example**:  **Use:**  

6

1

4

2

3

0

5

-1

-2

-3

|  |  |
| --- | --- |
| **x** | **f(x)** |
| 0 | 1 |
| 1 | -2 |
| 2 | -3 |
| 3 | -2 |
| 4 | 1 |
| 5 | 6 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **k**  6 6 1  4 1 4 0.25  3.75 0.0625 3.5 0.01786  3.7321 0.0003 3.4643 0.00009  3.7320 | **x** | **f(x)** | **∂f/∂x** | **f / (∂f/∂x)** |
| 1 | 5 |  |  |  |
| 2 |  |  |  |  |
| 3 |  |  |  |  |
| 4 |  |  |  |  |
| 5 |  |  |  |  |
| -2.75 1 -2.75  5.25 7.5625 6.5 1.1635  4.0865 1.3536 4.1731 0.3244  3.7622 0.1052 3.5243 0.0299  3.7323 |  |  |  |  |
| 1 | 2.5 |  |  |  |
| 2 |  |  |  |  |
| 3 |  |  |  |  |
| 4 |  |  |  |  |
| 5 |  |  |  |  |
| -2 -2 1  0 1 -4 -0.25  0.25 0.0625 -3.5 -0.01786  0.2679 0.00017 -3.4642 -0.00005  0.2679 |  |  |  |  |
| 1 | 1 |  |  |  |
| 2 |  |  |  |  |
| 3 |  |  |  |  |
| 4 |  |  |  |  |
| 5 |  |  |  |  |

# **Newton-Raphson for Four Bar**

**Given:** constants r1 r2 r3 r4  and variable  **Find:** 3 and 4

**Subject to:** 



**Use:**  generalized coordinates,  constraint functions

**Taylor series about estimate {q}k: **

 ****

**Desired constraint functions:** 

**Newton-Raphson equation:** 

**Example: r1 = 90 cm, r2 = 30 cm, r3 = 60 cm, r4 = 45 cm,  = 0°, 2 = 65**°

(3 = 13.151°,4 = 114.827° by manual geometric solution)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| k | [deg] | [cm] | [cm/rad] | [rad] | [deg] |
| 1 | 0  90 | -17.3215  -17.8108 | 1. 45   60 0 | -0.2968  -0.3849 | -17.0080  -22.0544 |
| 2 | 17.0080  112.0544 | -3.0488  3.0323 | -17.5503 41.7073  57.3758 16.8969 | 0.0662  -0.0453 | 3.7916  -2.5927 |
| 3 | 13.2164  114.6471 | -0.1444  0.0068 | -13.7178 40.9002  58.4108 18.7663 | 0.0011  -0.0032 | 0.0647  -0.1806 |
| 4 | 13.1517  114.8277 | -0.00013  0.00019 | -13.6518 40.8409  58.4263 18.8951 | 0.0000039  -0.0000018 | 0.00023  -0.00011 |
| 5 | 13.1515  114.8278 | 10-14  10-14 | det [q] =  -2644.1 |  |  |

 = 4 - 3 = 101.6763° -r3 r4 sin  = -2644.1

if  is the driver  **** 

if  is the driver  **** 

**Numerical Partial Derivatives**

1) use current generalized coordinates {q}

2) evaluate constraint functions {}

3) perturb one (and only one) qj by a small value  qj\* = qj + 

4) evaluate perturbed constraint functions {\*}

5) compute partial derivatives with respect to that qj 

6) column j of Jacobian 

7) remember to reset and use the original value for qj

8) repeat steps 3) through 7) for all j

**Example: r1 = 90 cm, r2 = 30 cm, r3 = 60 cm, r4 = 45 cm,  = 0°, 2 = 65**°













% test\_jac.m - evaluate Jacobian by numerical partial derivatives

% used for ME 581 web cutter

% HJSIII, 20.02.19

% hold estimates for generalized coordinates

nq = length(q);

qhold = q;

% evaluate constraints

wc\_phi

% hold constraints

phold = PHI;

% perturb one coordinate at a time

for iq = 1:nq,

q = qhold;

q(iq) = q(iq) + 0.01;

% change in constraints caused by coordinate perturbation is

% approximately equal to partial derivative

wc\_phi

jtest(:,iq) = ( PHI - phold ) / 0.01;

end

% reset coordinates and constraints

q = qhold;

wc\_phi

% bottom - test\_jac