APPENDIX A:

SUPPLEMENT TO CHAPTER 2

Appendix A: Supplement to chapter 2

- **Section A.1**: List of parameters and variables used for vehicle model
- Detailed derivations of some important equations used in chapter 2:
  - **Section A.2**: Deflection of equivalent suspension (Eq. 2.29)
  - **Section A.3**: Total moment acting on vehicle body due to axle i (Eq 2.43)
  - **Section A.4**: Accelerations of different wheels (Eq 2.45 & 2.46)
- **Section A.5**: Snyman's dynamic trajectory optimization method

A.1 List of parameters and variables used for vehicle model

a_i = Distances between axles i and $i+1$ in the initial prescribed position, $i = 1,2,3$

(c_i, x_i) = Parameter pairs for six piece-wise continuous linear representation of force element data, $i = 1,2,...,6$

c^y = Vertical position of centre of gravity

c^x = Horisontal position of centre of gravity

d_i = Horisontal length of the equivalent trailing arm of axle i, $i = 1,2,3,4$

d_i^w = Longitudinal distance d_i^w between the wheel centre of axle i, $i = 1,2,3,4$ and the centre of gravity as in the initial prescribed position

e_i = Vertical height of the equivalent trailing arm of axle i, $i = 1,2,3,4$

f_i^a = Force in the equivalent trailing arm of axle i, $i = 1,2,3,4$

f_i^b = Equivalent bump stop force on axle i, $i = 1,2,3,4$

f_i^d = Equivalent damper force on axle i, $i = 1,2,3,4$

f_i^e = Total equivalent suspension force on axle i, $i = 1,2,3,4$

f_i^s = Equivalent spring force on axle i, $i = 1,2,3,4$

f_i^w = Resultant tyre force on axle i, $i = 1,2,3,4$

F^a = Total sum of the components of the trailing arm forces in the direction of q_1

F_i^b = Actual spring force at a deflection of Δ_i^b on axle i, $i = 1,2,3,4$

F_i^d = Actual spring force at a deflection rate of $\dot{\Delta}_i^d$ on axle i, $i = 1,2,3,4$

F^s = Sum of the suspension forces

F_i^s = Actual spring force at a deflection of Δ_i^s on axle i, $i = 1,2,3,4$

F_i^{wd} = Wheel force due to the tyre deflection rate and damping on axle i, $i = 1,2,3,4$

F_i^{ws} = Wheel force due to the tyre deflection and stiffness on axle i, $i = 1,2,3,4$

h = Height of the centre of gravity with respect to the reference point 0,0 in the initial prescribed position

I = Pitch inertia of the sprung mass about its centre of gravity

L = Horisontal distance between the sprung mass centre of gravity and the centre of the front axle in the initial prescribed position

m_1 = Mass of the vehicle body (the sprung mass)

m_{i+1} = Mass of the unsprung mass associated with each axle i, $i = 1,2,3,4$

M_i = Moment applied on the vehicle body due to the forces at axle i, $i = 1,2,3,4$

n_i^w = Number of wheels per axle on axle i, $i = 1,2,3,4$

n_i^s = Number of springs on the axle on axle i, $i = 1,2,3,4$

n_i^b = Number of bump stops per axle on axle i ,i = 1,2,3,4

n_i^d = Number of dampers per axle on axle i ,i = 1,2,3,4

q_1 = Vertical displacement at the centre of gravity of the vehicle body

q_2 = Pitch displacement of the vehicle body

q_{i+2} = Vertical displacement at the wheel centre of axle i, i = 1,2,3,4

$\dot{q}_i$ = Velocity associated with q_i, $i = 1,2,...,6$

$\ddot{q}_i$ = Acceleration associated with q_i, $i = 1,2,...,6$

$\ddot{q}_i^a$ = Accelerations of the different wheels (perpendicular to the equivalent trailing arm) of axle i, i = 1,2,3,4

r_i = Rolling radius of the tyres on axle i ,i = 1,2,3,4

t = Simulation time in seconds

t_i = Length of the equivalent trailing arm on axle i, i = 1,2,3,4

v = Horisontal (forward) vehicle speed at the centre of gravity of the vehicle body

w_i^x = Horisontal position of the wheel centre on axle i, i = 1,2,3,4 as measured from the reference point

w_i^y = Vertical position of the wheel centre on axle i, i = 1,2,3,4 as measured from the reference point

β_i = Angle of the equivalent trailing arm on axle i, i = 1,2,3,4

δ_i = Deflection of the equivalent suspension on axle i =1,2,3,4

δ_i^s = Initial spring deflection on axle i, i = 1,2,3,4

δ_i^b = Initial bump stop deflection on axle i, i = 1,2,3,4

δ_i^t = Initial tyre deflection on axle i, i = 1,2,3,4

δ_i^w = Resultant tyre deflection on axle i, i = 1,2,3,4

$\dot{\delta}_i^w$ = Resultant tyre deflection rate on axle i, i = 1,2,3,4

δ^t = Integration time step

Δ_i^a = Intermediate distance used to calculate the wheel position w_i^x for axle i, i = 1,2,3,4

Δ_i^b = Intermediate distance used to calculate the wheel position w_i^x for axle i, i = 1,2,3,4

Δ_i^c = Intermediate distance used to calculate the wheel position w_i^x for axle i, i = 1,2,3,4

Δ_i^s = Deflection at the actual spring on axle i, i = 1,2,3,4

Δ_i^b = Deflection at the actual bump stop on axle i, i = 1,2,3,4

$\dot{\Delta}_i^s$ = Deflection rate at the actual damper on axle i, i = 1,2,3,4

θ = Pitch angle of vehicle on a specific time instance

φ_i = Angle of the resultant tyre force on axle i, i = 1,2,3,4 and the vertical

κ_i^s = Ratio between the actual spring deflection on axle i, i = 1,2,3,4 and that for the equivalent suspension

κ_i^b = Ratio between the actual bump stop deflection on axle i, $i = 1,2,3,4$ and that for the equivalent suspension

κ_i^d = Ratio between the actual damper deflection on axle i, $i = 1,2,3,4$ and that for the equivalent suspension

Λ_i = Tyre force factor on axle i, $i = 1,2,3,4$ determined by the shape of the tyre deflection

A.2 Deflection of equivalent suspension (eq. 2.29)

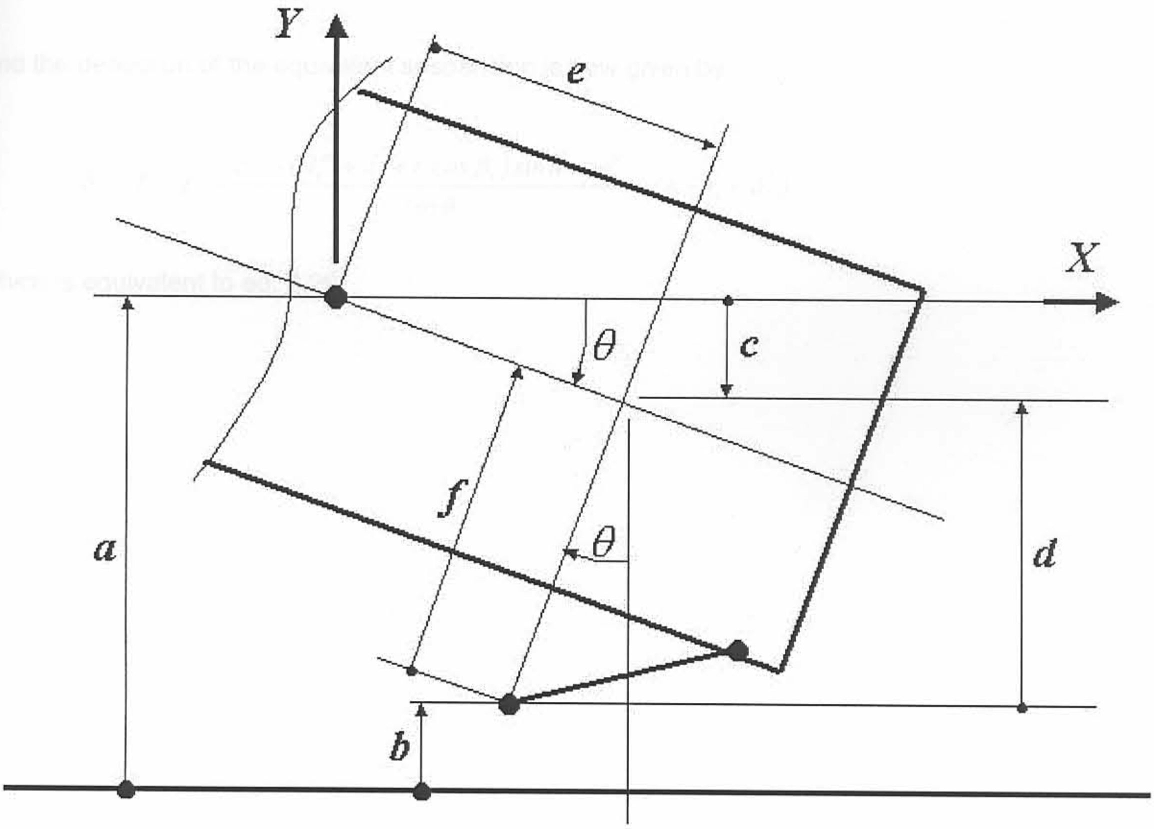

Figure A1: Schematic of vehicle body with one trailing arm for axle i

From figure A1 the value of f can be determined:

$$a = c^y$$

$$b = w^y$$

$$e = d_i^w + d_i + t_i \cos\beta_i \qquad (\text{remember } \cos\beta_i < 0)$$

$$c = e\sin\theta = (d_i^w + d_i + t_i\cos\beta_i)\sin\theta$$

$$d = a - c - b$$

$$f = \frac{d}{\cos\theta} = \frac{a-c-b}{\cos\theta} = \frac{c^y - (d_i^w + d_i + t_i\cos\beta_i)\sin\theta - w^y}{\cos\theta}$$

The original value for f (in the prescribed initial position), say f' is given by:

$$f' = h - (r_i + \delta_i^t)$$

and the deflection of the equivalent suspension is now given by:

$$\delta_i = f - f' = \frac{c^y - (d_i^w + d_i + t_i \cos\beta_i)\sin\theta - w_i^y}{\cos\theta} - (h - r_i - \delta_i^t)$$

which is equivalent to eq. 2.29.

A.3 Total moment acting on vehicle body due to axle *i* (eq. 2.43)

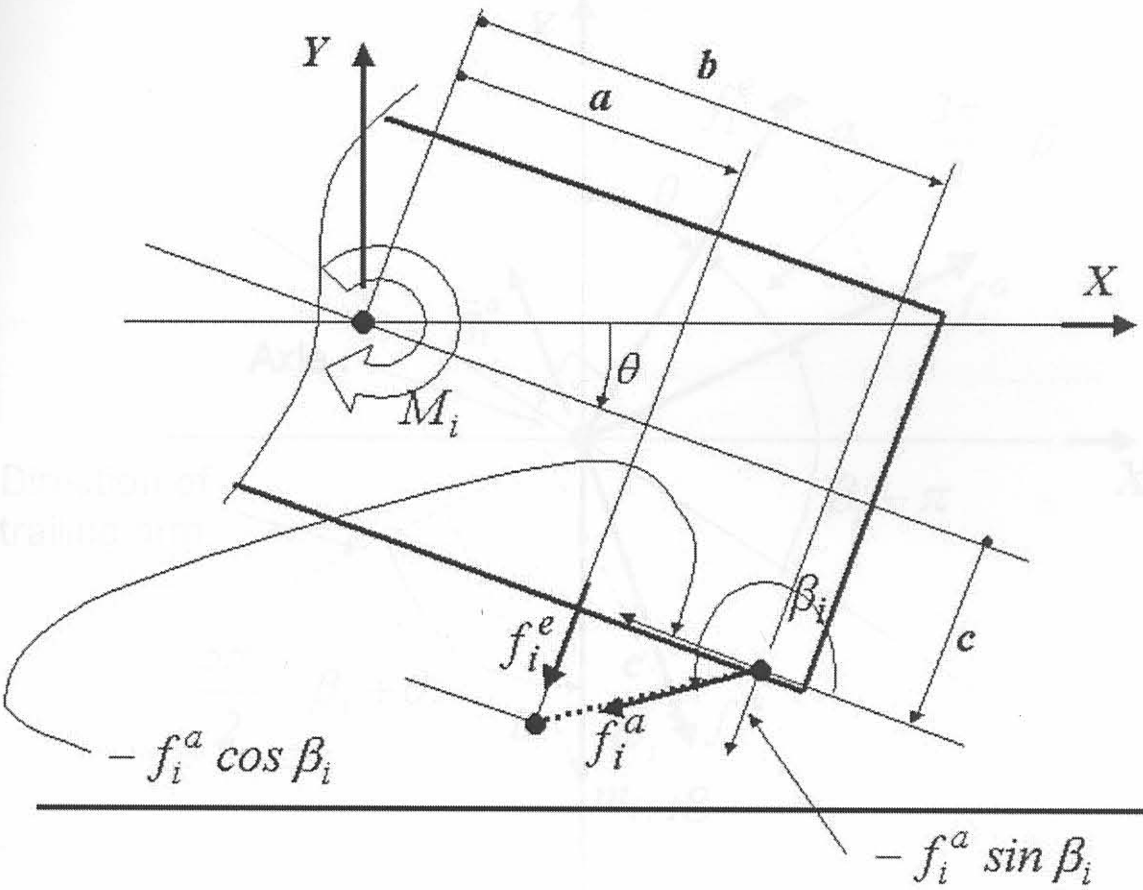

Figure A2: Schematic of vehicle body with one trailing arm for axle i

From figure A2 the moment acting on the vehicle can be determined as:

$$a = d_i^w + d_i + t_i \cos\beta_i \qquad (remember\ \cos\beta_i < 0)$$

$$b = d_i^w + d_i$$

$$c = h - (r_i + \delta_i^t + e_i)$$

$$M_i = af_i^e + b(-f_i^a \sin\beta_i) + c(-f_i^a \cos\beta_i)$$

$$\Rightarrow M_i = f_i^e(d_i^w + d_i + t_i \cos\beta_i) - f_i^a(d_i^w + d_i)\sin\beta_i - f_i^a(h - r_i - \delta_i^t - e_i)\cos\beta_i$$

which is equivalent to eq. 2.43

A.4 Acceleration of axle center for axle i (eq. 2.45 & 2.46)

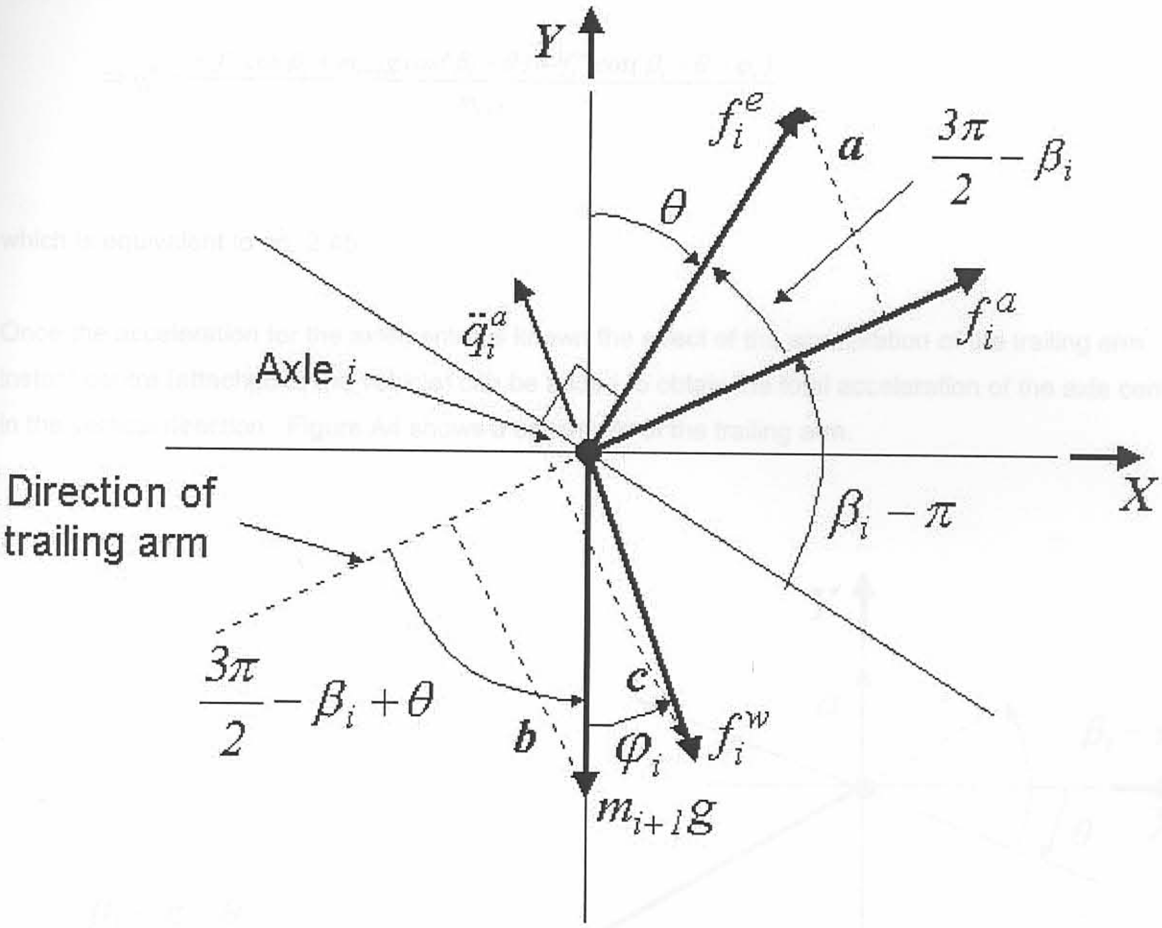

Figure A3: Schematic of axle i

From figure A3 the forces a,b and c is given by:

$$a = f_i^e \, sin(\frac{3\pi}{2} - \beta_i) = -f_i^e \, cos\, \beta_i$$

$$b = m_{i+1} g \, sin(\frac{3\pi}{2} - \beta_i + \theta) = -m_{i+1} g \, cos(-\beta_i + \theta) = -m_{i+1} g \, cos(\beta_i - \theta)$$

$$c = f_i^w \, sin(\frac{3\pi}{2} - \beta_i + \theta + \varphi_i) = -f_i^w \, cos(\beta_i - \theta - \varphi_i)$$

According to Newton's second law the acceleration of the axle centre (perpendicular to the trailing arm) is now given by:

$$\ddot{q}_i^a = \frac{a+b+c}{m_{i+1}}$$

$$\Rightarrow \ddot{q}_i^a = \frac{-f_i^e \cos\beta_i + m_{i+1} g \cos(\beta_i - \theta) + f_i^w \cos(\beta_i - \theta - \varphi_i)}{m_{i+1}}$$

which is equivalent to eq. 2.45.

Once the acceleration for the axle centre is known the effect of the acceleration of the trailing arm instant centre (attached to the vehicle) can be added to obtain the total acceleration of the axle centre in the vertical direction. Figure A4 shows a schematic of the trailing arm.

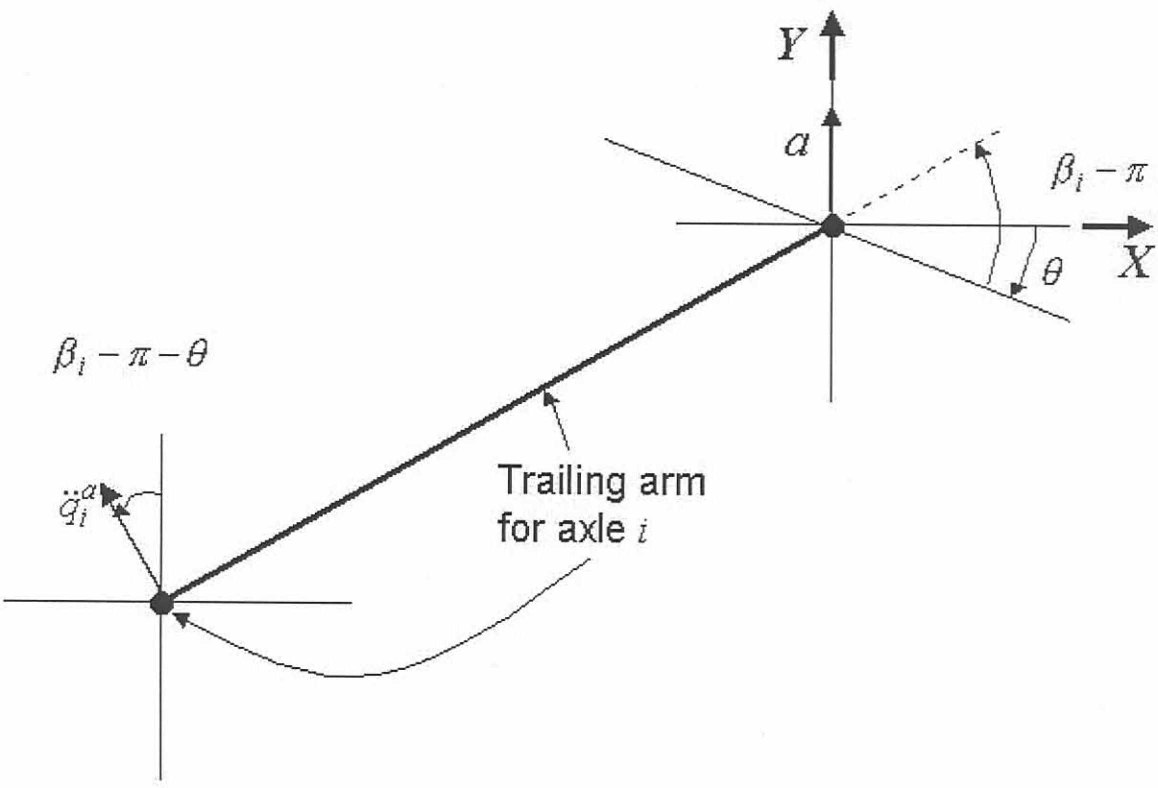

Figure A4: Schematic of trailing arm for axle i

The acceleration of the trailing arm centre (attached to the vehicle), due to the vehicle vertical acceleration and pitch acceleration is given by (also refer to figure A3):

$$a=\ddot{q}_1-\ddot{q}_2[(d_i^w+d_i)\cos\theta-(h-r_i-\delta_i^t-e_i)\sin\theta]$$

and the vertical acceleration of the trailing arm centre due to $\ddot{q}_i^a$ is given by:

$$b=\ddot{q}_i^a\cos(\beta_i-\theta-\pi)=-\ddot{q}_i^a\cos(\beta_i-\theta)$$

The total acceleration of the wheel centre (vertically upwards) is then given by:

$$\ddot{q}_{i+2}=b+a=-\ddot{q}_i^a\cos(\beta_i-\theta)+\ddot{q}_1-\ddot{q}_2[(d_i^w+d_i)\cos\theta-(h-r_i-\delta_i^t-e_i)\sin\theta]$$

which is eq. 2.46

A.5 Snyman's dynamic trajectory optimization method

Background

The dynamic trajectory method (also called the "leap-frog" method) for the unconstrained minimization of a scalar function $f(x)$ of n real variables represented by the vector $x=(x_1,x_2,\ldots,x_n)^T$ was originally proposed by Snyman [A1,A2]. The original algorithm has recently been modified to handle constraints by means of a penalty function formulation. (Snyman et al [A3,A4]). The method possesses the following characteristics:

- It uses only function *gradient* information $\nabla f(x)$.
- *No* explicit *line searches* are performed.
- It is extremely *robust* and handles steep valleys and discontinuities in functions and gradients with ease.
- The algorithm seeks a *low local minimum* and can therefore be used as a basic component in a methodology for global optimization.
- It is not as efficient as classical methods on smooth and near-quadratic functions.

Basic dynamic model

The algorithm is modelled on the motion of a particle of unit mass in a n-dimensional conservative force field with potential energy at x given by $f(x)$. At x, the force on the particle is given by

$$a = \ddot{x} = -\nabla f(x) \tag{A.1}$$

from which it follows that for the time interval [0,t]

$$\begin{aligned} \tfrac{1}{2}\|\dot{x}(t)\|^2 - \tfrac{1}{2}\|\dot{x}(0)\|^2 &= f(x(0)) - f(x(t)) \\ T(t) - T(0) &= f(0) - f(t) \end{aligned} \tag{A.2}$$

or

$$f(t) + T(t) = \text{constant \{conservation of energy\}}$$

Note that since $\Delta f = -\Delta T$ as long as T *increases* f *decreases. This forms the basis of the dynamic algorithm.*

LFOP: Basic algorithm for unconstrained problems

Given $f(\boldsymbol{x})$ and a starting point $\boldsymbol{x}(0)=\boldsymbol{x}^0$

- Compute the dynamic trajectory by solving the initial value problem (IVP):

$$\ddot{\boldsymbol{x}}(t) = -\nabla f(\boldsymbol{x}(t)) \qquad \textbf{(A.3)}$$
$$\dot{\boldsymbol{x}}(0) = \boldsymbol{0}, \quad \boldsymbol{x}(0) = \boldsymbol{x}^0$$

- Monitor $\boldsymbol{v}(t) = \dot{\boldsymbol{x}}(t)$. Clearly as long as $T = \frac{1}{2}\|\boldsymbol{v}(t)\|^2$ increases $f(\boldsymbol{x}(\mathrm{t}))$ decreases as required.
- When $\|\boldsymbol{v}(t)\|$ decreases apply some interfering strategy to extract energy and thereby increase the likelihood of descent.
- In practice a numerical integration "leap-frog" scheme is used to integrate the IVP **(A.3)** Compute for k=0,1,2,… and time step Δt

$$\boldsymbol{x}^{k+1} = \boldsymbol{x}^k + \boldsymbol{v}^k \Delta t \qquad \textbf{(A.4)}$$
$$\boldsymbol{v}^{k+1} = \boldsymbol{v}^k + \boldsymbol{a}^{k+1}\Delta t$$

where $\boldsymbol{a}^k = -\nabla f(\boldsymbol{x}^k), \quad \boldsymbol{v}^0 = \frac{1}{2}\boldsymbol{a}^0 \Delta t$

- A typical interfering strategy is

If $\|\boldsymbol{v}^{k+1}\| \geq \|\boldsymbol{v}^k\|$ continue

else

set $\boldsymbol{v}^k = \dfrac{\boldsymbol{v}^{k+1} + \boldsymbol{v}^k}{4}, \quad \boldsymbol{x}^k = \dfrac{\boldsymbol{x}^{k+1} + \boldsymbol{x}^k}{2}$ **(A.5)**

compute new $\boldsymbol{v}^{k+1}$ and continue.

- Further heuristics are used to determine an initial Δt, to allow for the magnification and reduction of Δt, and to control the step size.

LFOPC: Modification for constrained problems

Constrained optimization problems are solved by the application, in three phases, of LFOP to a penalty function formulation of the problem **[A3,A4]**. Given a function $f(\boldsymbol{x})$, with equality constraints h_i=0 (i=1,2,…,r) and inequality constraints g_j≤0 (j=1,2,…,m) and penalty parameter μ>>0, the penalty function problem is to minimize

$$P(\boldsymbol{x},\mu) = f(\boldsymbol{x}) + \sum_{i=1}^{r} \mu h_i^2(\boldsymbol{x}) + \sum_{j=1}^{m} \beta_j g_j^2(\boldsymbol{x}) \qquad \textbf{(A.6)}$$

$$\text{where } \beta_j = \begin{cases} 0 \text{ if } g_j(\boldsymbol{x}) \le 0 \\ \mu \text{ if } g_j(\boldsymbol{x}) > 0 \end{cases}$$

Phase 0: Given some $\boldsymbol{x}^0$, then with the overall penalty parameter $\mu = \mu_0 (= 10^2)$ apply LFOP to $P(\boldsymbol{x},\mu_0)$ to give $\boldsymbol{x}^*(\mu_0)$

Phase 1: With $\boldsymbol{x}^0 = \boldsymbol{x}^*(\mu_0)$, $\mu = \mu_1 (= 10^4)$ apply LFOP to $P(\boldsymbol{x},\mu_1)$ to give $\boldsymbol{x}^*(\mu_1)$ and identify active constraints $i_a = 1,2,\ldots,n_a$; $g_{i_a}(\boldsymbol{x}^*(\mu_1)) > 0$

Phase 2: With $\boldsymbol{x}^0 = \boldsymbol{x}^*(\mu_1)$, use LFOP to minimize

$$P_a(\boldsymbol{x},\mu_1) = \sum_{i=1}^{r} \mu_1 h_i^2(\boldsymbol{x}) + \sum_{i_a=1}^{n_a} \mu_1 g_{i_a}^2(\boldsymbol{x}) \qquad \textbf{(A.7)}$$

to give $\boldsymbol{x}^*$.

REFERENCES

A1. J.A. Snyman, *'A new and dynamic method for unconstrained minimization'*, Appl. Math. Modeling. **6**, pp 449-462 (1982).

A2. J.A. Snyman, *'An improved version of the original leap-frog method for unconstrained minimization'*, Appl. Math. Modeling. **7**, pp 216-218 (1983).

A3. J.A. Snyman, W.J. Roux and N. Stander, *'A dynamic penalty function method for the solution of structural optimization problems'*, Appl. Math. Modeling. **180** (15), pp 371-386 (1994).

A4. J.A. Snyman, *'The LFOPC leap-frog algorithm for constrained optimization'*, Computers and Mathematics with Applications, **40**, pp 1085-1096 (2000).

APPENDIX B:

Compact disc with Vehsim2d (demonstration version)

Appendix B: Compact disc with Vehsim2d (demonstration version)

Note:

Vehsim2d uses a graphical user interface (GUI) operating in a ***Microsoft Windows 95/98/2K/NT*** environment. It was developed using Microsoft Visual Basic 6.0 (Enterprise edition).

Installation instructions:

Run the program Vehsim2d_Setup.exe that can be found on the CD. This will run the setup program for installation of the Vehsim2d/LFOPC program and the associated help files.