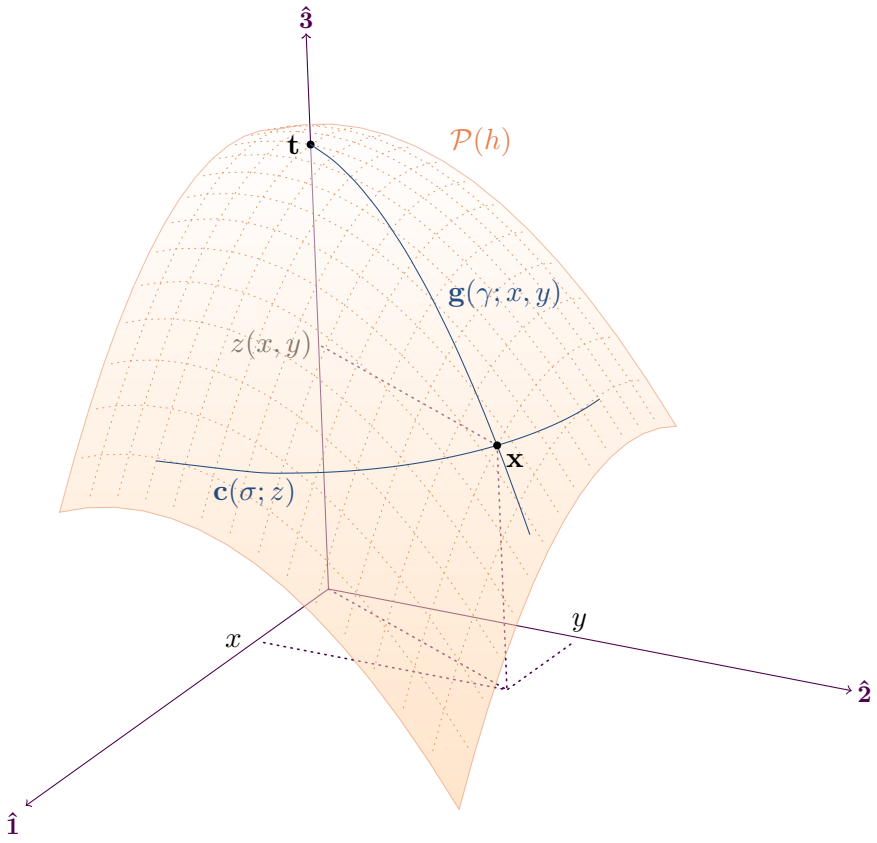
A Study of Contour and Gradient Paths on Surfaces Embedded in $\mathbb{R}^3$

Paul Kotschy

11 June 2016

Compiled on March 16, 2025



Abstract

A STUDY OF CONTOUR AND GRADIENT PATHS on surfaces embedded in $\mathbb{R}^3$ is presented.[1] An interesting formula is introduced for the gradient path passing over any point of interest in the embedded surface. A systematic procedure is introduced for calculating both contour and gradient paths. The surface itself, the contour path, and the gradient path exist as geometrical objects in their own right, independent of the choice of coordinates. However, they admit a specific set of coordinates which seem "natural" to the surface. This is studied. The commutitivity of contour path and gradient path traversal for a flat plane and for an inverted parabola is analysed.

[1] paul.kotschy@gmail.com

Contents

1 Introduction

Part of a surface $\mathcal{S}$ embedded in $\mathbb{R}^3$ may be specified with the functional association

$$z : (x, y) \mapsto z(x, y), \quad x, y, z(x, y) \in \mathbb{R} \tag{1}$$

The surface itself is a subset of the $\mathbb{R}^3$ vector space, and is given by

$$\mathcal{S} = \{(x, y, z) \,|\, z = (x, y))\}$$

As a member of $\mathcal{S}$, the triple $(x, y, z(x, y))$ is called a point on $\mathcal{S}$, and may be represented geometrically by a position vector $\mathbf{x} \in \mathbb{R}^3$ derived from the $(x, y, z(x, y))$ triple as the linear combination

$$\mathbf{x}(x, y) = x\hat{\mathbf{1}} + y\hat{\mathbf{2}} + z(x, y)\hat{\mathbf{3}} \tag{2}$$

where $\{\hat{\mathbf{1}}, \hat{\mathbf{2}}, \hat{\mathbf{3}}\}$ is the usual orthonormal vector basis for $\mathbb{R}^3$.

As either of the x or y in the (x, y) pair in (1) varies continuously, it traces out a path (or curve) over $\mathcal{S}$ specified by (2). These paths shall be of primary interest in this work.

Two tractable examples of such embedded surface sets are the flat plane and the inverted paraboloid centred at the origin. Two specific instances of these are:

$$\begin{aligned} \mathcal{L}(h) =& \{(x, y, z) \,|\, z = h - x - y\} \\ \mathcal{P}(h) =& \{(x, y, z) \,|\, z = \frac{1}{h}\left(h^2 - x^2 - y^2\right)\} \end{aligned} \tag{3}$$

for some parameter $h \in \mathbb{R}$. In this work, I shall use $\mathcal{L}(h)$ and $\mathcal{P}(h)$ as case studies. Parts of a flat plane and an inverted paraboloid are depicted in Figure 1 (page 6) and Figure 2 (page 9).

In such depictions, x and y are obvious candidates to coordinatise $\mathcal{S}$. But they are merely coordinates. The surfaces exist as entities in their own right, independently of the choice of coordinates. What is important is what defines a surface, namely, a specification of the association $z : (x, y) \mapsto z(x, y)$. For example, suppose while working with $\mathcal{S}$, it is convenient to use s and t as coordinates, where s could stand for spatial path separation, say, and t for time. Then a surface embedded in $\mathbb{R}^3$ is

$$\{(x, y, z) \,|\, z = z(x, y),\ x = x(s, t),\ y = y(s, t)\}$$

and the geometrical representation becomes

$$\mathbf{x}(s, t) \ = \ \mathbf{x}(x(s, t), y(s, t)) \ = \ x(s, t)\hat{\mathbf{1}} \ + \ y(s, t)\hat{\mathbf{2}} \ + \ z(x(s, t), y(s, t))\hat{\mathbf{3}}$$

where the association

$$(s, t) \mapsto (x(s, t), y(s, t))$$

is called the coordinate transformation from s and t to x and y.

Does the surface offer any coordinate systems which seem natural for the surface? Suppose you are hiking in mountainous terrain. Then a natural inclination might be to find a route which either maintains a constant height, or which offers the most direct ascent or descent. The former route would follow a *contour path*, while the latter route would follow a *gradient path*.

2 Contour path

If a point (x_0, y_0, z_0) lies on our surface $\mathcal{S}$, then it must satisfy (1):

$$z : (x_0, y_0) \mapsto z(x_0, y_0) = z_0$$

The corresponding position vector is

$$\mathbf{x}_0 \ = \ \mathbf{x}(x_0, y_0) \ = \ x_0\hat{\mathbf{1}} + y_0\hat{\mathbf{2}} + z(x_0, y_0)\hat{\mathbf{3}} \ = \ x_0\hat{\mathbf{1}} + y_0\hat{\mathbf{2}} + z_0\hat{\mathbf{3}}$$

But in general, there are other points (x, y) which satisfy $z(x, y) = z_0$. Therefore, we may meaningfully define the *contour at* z_0 as the set

$$\mathcal{C}(z_0) = \{(x, y, z_0) \,|\, z(x, y) = z_0\}$$

The embedded surface $\mathcal{S}$ is a two-dimensional object because two independent parameters are needed for its specification. But the contour $\mathcal{C}(z_0)$ is one-dimensional because the condition $z(x, y) = z_0$ affirms a relation between x and y. So if we choose σ as the single independent variable, say, then σ traces out a path specified by the position vector $\mathbf{c} \in \mathbb{R}^3$ given by

$$\mathbf{c}(\sigma; z_0) = c_1(\sigma; z_0)\hat{\mathbf{1}} + c_2(\sigma; z_0)\hat{\mathbf{2}} + z_0\hat{\mathbf{3}} \tag{4}$$

I shall call this the z_0-contour path of $\mathcal{S}$. For example, suppose the point (x_0, y_0, z_0) lies on our flat plane $\mathcal{L}(h)$ (Eq. (3)), then (x_0, y_0, z_0) must satisfy

$$x_0 + y_0 = h - z_0$$

So $\mathcal{L}(h)$'s z_0-contour is the set $\{(x, y, z_0) \,|\, x + y = h - z_0)\}$, and $\mathcal{L}(h)$'s z_0-contour path may be specified by the position vector

$$\mathbf{c}(\sigma; z_0) = c_1(\sigma; z_0)\hat{\mathbf{1}} \,+\, (h - z_0 - c_1(\sigma; z_0))\,\hat{\mathbf{2}} \,+\, z_0\hat{\mathbf{3}}$$

Equation (4) specifies a continuous variation of the position vector $\mathbf{c}(\sigma; z_0)$ with σ. This variation of a vector quantity along a path is perhaps the most natural one.

Another vector variation is the rate of change of the position vector, aligned with the direction of travel along the contour over $\mathcal{S}$. For any arbitrary path over $\mathcal{S}$, coordinatised by s, say, the path's tangent vector is

$$\begin{aligned}\frac{\mathrm{d}\mathbf{x}(s)}{\mathrm{d}s} \;=\; \frac{\mathrm{d}}{\mathrm{d}s}\mathbf{x}(x(s), y(s)) \;&=\; \frac{\mathrm{d}x}{\mathrm{d}s}\hat{\mathbf{1}} + \frac{\mathrm{d}y}{\mathrm{d}s}\hat{\mathbf{2}} + \left(\frac{\partial z}{\partial x}\frac{\mathrm{d}x}{\mathrm{d}s} + \frac{\partial z}{\partial y}\frac{\mathrm{d}y}{\mathrm{d}s}\right)\hat{\mathbf{3}} \\ &=\; \frac{\mathrm{d}x}{\mathrm{d}s}\hat{\mathbf{1}} + \frac{\mathrm{d}y}{\mathrm{d}s}\hat{\mathbf{2}} + \nabla_{(x,y)}z(x, y) \cdot \left(\frac{\mathrm{d}x}{\mathrm{d}s}\hat{\mathbf{1}} + \frac{\mathrm{d}y}{\mathrm{d}s}\hat{\mathbf{2}}\right)\hat{\mathbf{3}}\end{aligned}$$

From (4), the $\hat{\mathbf{3}}$-component of the tangent vector along the z_0-contour path must vanish, giving

$$\frac{\mathrm{d}\mathbf{c}(\sigma; z_0)}{\mathrm{d}\sigma} \;=\; \frac{\mathrm{d}c_1(\sigma; z_0)}{\mathrm{d}\sigma}\hat{\mathbf{1}} + \frac{\mathrm{d}c_2(\sigma; z_0)}{\mathrm{d}\sigma}\hat{\mathbf{2}} \tag{5}$$

3 Gradient path

The two vector quantities (4) and (5) pertain to the z_0-contour path. There is another path over our embedded surface $\mathcal{S}$, now parametrised by γ, say, passing through the position $\mathbf{x}_0$, and whose tangent vector is orthogonal to the z_0-contour path's tangent vector. Call this path the $\mathbf{x}_0$-gradient path $\mathbf{g}$, and write

$$\mathbf{g}(\gamma; x_0, y_0) = g_1(\gamma; x_0, y_0)\hat{\mathbf{1}} + g_2(\gamma; x_0, y_0)\hat{\mathbf{2}} + z(g_1, g_2)\hat{\mathbf{3}}$$

for some as yet unspecified g_1, g_2. Once again, a tangent vector along $\mathbf{g}$ is

$$\frac{\mathrm{d}\mathbf{g}(\gamma; x_0, y_0)}{\mathrm{d}\gamma} \;=\; \frac{\mathrm{d}g_1}{\mathrm{d}\gamma}\hat{\mathbf{1}} + \frac{\mathrm{d}g_2}{\mathrm{d}\gamma}\hat{\mathbf{2}} + \frac{\mathrm{d}z(g_1, g_2)}{\mathrm{d}\gamma}\hat{\mathbf{3}}$$

Now consider another vector field over $\mathcal{S}$ defined by:

$$\mathbf{G}(x, y) \;=\; \frac{\partial z}{\partial x}\hat{\mathbf{1}} + \frac{\partial z}{\partial y}\hat{\mathbf{2}} + G_3(x, y)\hat{\mathbf{3}}$$

for some as yet unspecified $G_3(x, y)$, and consider its value at the point (x_0, y_0, z_0) on $\mathcal{S}$. Then

$$\begin{aligned}\mathbf{G}(x_0, y_0) \cdot \frac{\mathrm{d}\mathbf{c}(\sigma_0; z_0)}{\mathrm{d}\sigma} &= \frac{\partial z(x_0, y_0)}{\partial x}\frac{\mathrm{d}c_1(\sigma_0; z_0)}{\mathrm{d}\sigma} + \frac{\partial z(x_0, y_0)}{\partial y}\frac{\mathrm{d}c_2(\sigma_0; z_0)}{\mathrm{d}\sigma} \\ &= \frac{\mathrm{d}z(c_1(\sigma_0; z_0), c_2(\sigma_0; z_0))}{\mathrm{d}\sigma} \\ &= \frac{\mathrm{d}z_0}{\mathrm{d}\sigma} \\ &= 0\end{aligned}$$

This inner product vanishes because σ parametrises the $\mathbf{c}(\sigma; z_0)$ contour path, and on that path, $z = z_0$. So $\mathbf{G}$ is orthogonal to $\mathbf{c}$ at the position $\mathbf{x}_0$. But $\mathbf{G}$ is not yet necessarily tangential to $\mathbf{g}$ at $\mathbf{x}_0$. For it to be tangential to $\mathbf{g}$, it would have to align with $\mathrm{d}\mathbf{g}/\mathrm{d}\gamma$. The alignment is obviously satisfied if we define $\mathbf{g}$ by equating $\mathbf{G}$ with $\mathrm{d}\mathbf{g}/\mathrm{d}\gamma$ at $\mathbf{x}_0$. That is:

$$\begin{aligned}\frac{\mathrm{d}g_1(\gamma_0, x_0, y_0)}{\mathrm{d}\gamma} &= \frac{\partial z(x_0, y_0)}{\partial x} \\ \frac{\mathrm{d}g_2(\gamma_0, x_0, y_0)}{\mathrm{d}\gamma} &= \frac{\partial z(x_0, y_0)}{\partial y} \\ \frac{\partial z(x_0, y_0)}{\partial x}\frac{\mathrm{d}g_1(\gamma_0, x_0, y_0)}{\mathrm{d}\gamma} + \frac{\partial z(x_0, y_0)}{\partial y}\frac{\mathrm{d}g_2(\gamma_0, x_0, y_0)}{\mathrm{d}\gamma} &= G_3(x_0, y_0)\end{aligned}$$

giving

$$G_3(x_0, y_0) = \left[\nabla_{(x,y)} z(x_0, y_0)\right]^2$$

So a tangent vector to our desired path $\mathbf{g}(\gamma; x, y)$ at $\mathbf{x}_0$ is

$$\frac{\mathrm{d}\mathbf{g}(\gamma_0; x_0, y_0)}{\mathrm{d}\gamma} = \frac{\partial z(x_0, y_0)}{\partial x}\,\hat{\mathbf{1}} + \frac{\partial z(x_0, y_0)}{\partial y}\,\hat{\mathbf{2}} + \left[\nabla_{(x,y)} z(x_0, y_0)\right]^2 \hat{\mathbf{3}}$$

Since the point (x_0, y_0, z_0) on $\mathcal{S}$ is arbitrarily chosen, the result also holds for the point (x, y, z) on $\mathcal{S}$. We may therefore drop the "$_0$" subscript.

To summarise, at any sufficiently well-behaved point (x, y, z) on the surface $\mathcal{S}$, the surface provides two natural paths. The paths are independent of the choice of coordinate system, and both are specified by two corresponding orthogonal tangent vectors:

1. z-contour path parametrised by σ, say:

$$\mathbf{c}(\sigma; z) = c_1(\sigma; z)\hat{\mathbf{1}} + c_2(\sigma; z)\hat{\mathbf{2}} + z\hat{\mathbf{3}} \tag{6}$$

2. Tangent vector of z-contour path:

$$\frac{\mathrm{d}\mathbf{c}(\sigma; z)}{\mathrm{d}\sigma} = \frac{\mathrm{d}c_1(\sigma; z)}{\mathrm{d}\sigma}\hat{\mathbf{1}} + \frac{\mathrm{d}c_2(\sigma; z)}{\mathrm{d}\sigma}\hat{\mathbf{2}} + 0\hat{\mathbf{3}} \tag{7}$$

3. $\mathbf{x}$-gradient path parametrised by γ, say:

$$\mathbf{g}(\gamma; x, y) = g_1(\gamma; x, y)\hat{\mathbf{1}} + g_2(\gamma; x, y)\hat{\mathbf{2}} + z(g_1, g_2)\hat{\mathbf{3}} \tag{8}$$

4. Tangent vector of $\mathbf{x}$-gradient path:

$$\begin{aligned}\frac{\mathrm{d}\mathbf{g}(\gamma; x, y)}{\mathrm{d}\gamma} &= \frac{\mathrm{d}g_1(\gamma; x, y)}{\mathrm{d}\gamma}\hat{\mathbf{1}} + \frac{\mathrm{d}g_2(\gamma; x, y)}{\mathrm{d}\gamma}\hat{\mathbf{2}} + \frac{\mathrm{d}z(g_1, g_2)}{\mathrm{d}\gamma}\hat{\mathbf{3}} \\ &= \frac{\partial z(x, y)}{\partial x}\,\hat{\mathbf{1}} + \frac{\partial z}{\partial y}\,\hat{\mathbf{2}} + \left[\nabla_{(x,y)} z(x, y)\right]^2 \hat{\mathbf{3}}\end{aligned} \tag{9}$$

In Sections 6 and 7, this summary will be used to calculate contour and gradient paths for $\mathcal{L}(h)$ and $\mathcal{P}(h)$.

4 Natural coordinates

As discussed earlier (page 3, and Eqs. (6) and (8)), the z-contour path $\mathbf{c}(\sigma; z)$ passes through the point (x, y, z) on $\mathcal{S}$. While tracing out the path, the $\hat{\mathbf{3}}$-component of our position vector remains constant at z, and our direction of travel aligns with the tangent vector $\mathrm{d}\mathbf{c}(\sigma; z)/\mathrm{d}\sigma$. The $\mathbf{x}$-gradient path $\mathbf{g}(\gamma; x, y)$ also passes through (x, y, z), and is the path of most direct ascent or descent with respect to the $\hat{\mathbf{3}}$-component of our position vector $\mathbf{x}$ on $\mathcal{S}$. And while tracing out that path, our direction of travel aligns with $\mathrm{d}\mathbf{g}(\gamma; x, y)/\mathrm{d}\gamma$.

We wish now to coordinatise our "travel experience" on $\mathcal{S}$ with coordinates which are associated with these two natural paths. The procedure is to define the three normalised orthogonal vectors at the position $\mathbf{x}$:

$$\begin{aligned} \hat{\mathbf{c}}(\mathbf{x}) &= \frac{\mathrm{d}\mathbf{c}(\sigma_0; z)}{\mathrm{d}\sigma} \Big/ \left|\frac{\mathrm{d}\mathbf{c}(\sigma_0; z)}{\mathrm{d}\sigma}\right| \\ \hat{\mathbf{g}}(\mathbf{x}) &= \frac{\mathrm{d}\mathbf{g}(\gamma_0; z)}{\mathrm{d}\gamma} \Big/ \left|\frac{\mathrm{d}\mathbf{g}(\gamma_0; z)}{\mathrm{d}\gamma}\right| \\ \hat{\mathbf{n}}(\mathbf{x}) &= \hat{\mathbf{c}}(\mathbf{x}) \times \hat{\mathbf{g}}(\mathbf{x}) \end{aligned} \tag{10}$$

and then to express $\mathbf{x}$ under the $\{\hat{\mathbf{c}}, \hat{\mathbf{g}}, \hat{\mathbf{n}}\}$ orthornomal vector basis, and finally to compute $\mathbf{x}$'s coordinates relative to this basis. This will be done for $\mathcal{L}(h)$ and $\mathcal{P}(h)$ in Sections 6 and 7 below.

5 Commutivity of path traversals

As will be shown later, in the Euclidean $\hat{\mathbf{1}}\hat{\mathbf{2}}$ coordinate plane, a traversal along a contour path commutes with a traversal along a gradient path. This commutivity does not carry over to non-Euclidean surfaces.

To demonstrate this, introduce traversal length l as a natural path parameter. Suppose that some path over our embedded surface $\mathcal{S}$ is parametrised by τ, i.e., $\mathbf{x} = \mathbf{x}(\tau)$ for $\tau \in \mathbb{R}$. Then a corresponding tangent vector along the path is $\mathrm{d}\mathbf{x}(\tau)/\mathrm{d}\tau$. The length of traversal over a portion of the path is then

$$l(\tau_0, \tau) = \int_{\tau_0}^{\tau} \left|\frac{\mathrm{d}\mathbf{x}(\bar{\tau})}{\mathrm{d}\bar{\tau}}\right| \mathrm{d}\bar{\tau} \tag{11}$$

This provides us with traversal length as a function of the arbitrary parameter τ and some initial value τ_0. In principle, this relationship can be inverted to give τ as a function of l. The path over $\mathcal{S}$ can then be parametrised with l as $\mathbf{x} = \mathbf{x}(\tau(l)) = \mathbf{x}(l)$.

Next, define the two coordinate mappings over $\mathcal{S}$:

$$\begin{aligned} \mathcal{C}(l) : \mathbf{x} \mapsto \mathcal{C}(l)\mathbf{x} = \mathbf{c}(l; z) \\ \mathcal{G}(m) : \mathbf{x} \mapsto \mathcal{G}(m)\mathbf{x} = \mathbf{g}(m; x, y) \end{aligned} \tag{12}$$

Here, the contour path $\mathbf{c}$ and gradient path $\mathbf{g}$ passing through the position $\mathbf{x}$ are now parametrised by lengths l and m, respectively. I.e., $\mathbf{c}(l; z) = \mathbf{c}(\sigma(l); z)$ (Eq. (6)) and $\mathbf{g}(m; x, y) = \mathbf{g}(\gamma(m); x, y)$ (Eq. (8)). The commutivity of contour and gradient path traversals may then be analysed by calculating the difference

$$(\mathcal{G}(m)\mathcal{C}(l) - \mathcal{C}(l)\mathcal{G}(m))\,\mathbf{x}$$

This will be done in Sections 6 and 7.

6 Embedded flat plane

Given its simplicity, the embedded flat plane $\mathcal{L}(h)$ (Eq. (3)) offers a useful—albeit, mundane—starting point for studying contour and gradient paths. Part of a typical flat plane is shown in Figure 1.

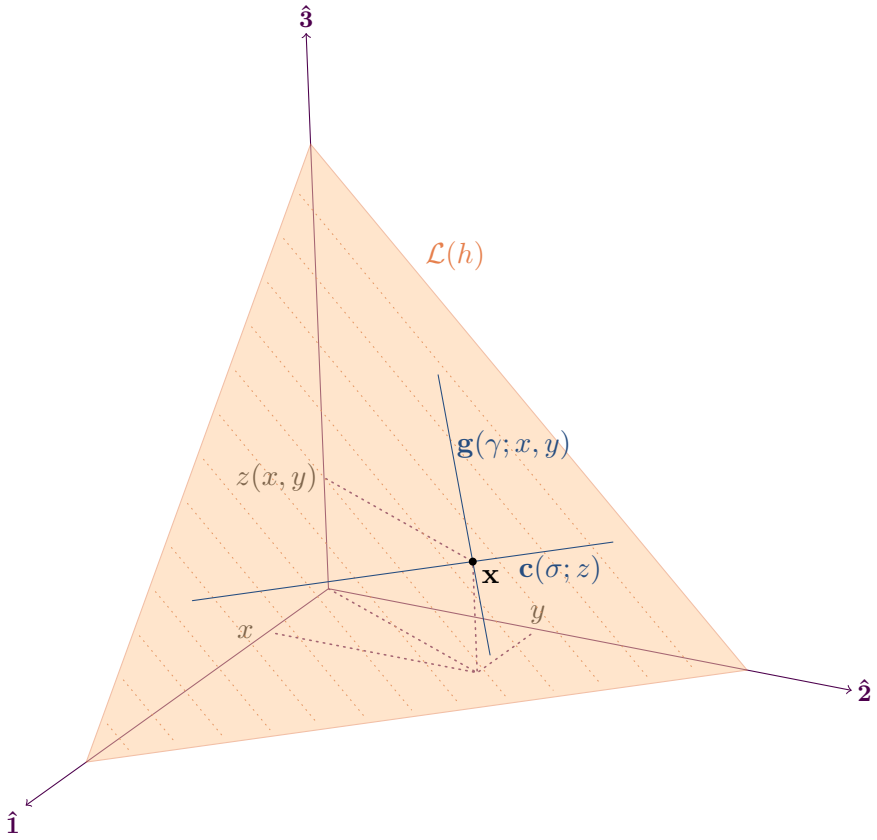


Figure 1: Flat plane $\mathcal{L}(h)$ specified by (3), showing a contour path $\mathbf{c}(\sigma; z)$ and a gradient path $\mathbf{g}(\gamma; x, y)$ intersecting orthogonally at the point $\mathbf{x}$ on $\mathcal{L}(h)$. Both $\mathbf{c}$ and $\mathbf{g}$ were calculated using (6), (7), (8) and (9).

Contour path. From (6), a z-contour path on $\mathcal{L}(h)$ is, say,

$$\mathbf{c}(\sigma; z) \;=\; (x - \sigma)\hat{\mathbf{1}} + \mu(\sigma)\hat{\mathbf{2}} + (h - (x - \sigma) - \mu(\sigma))\,\hat{\mathbf{3}}$$

The $\hat{\mathbf{1}}$-component is parametrised as $(x - \sigma)$ so that the $\hat{\mathbf{1}}$-component decreases with increasing σ, in keeping with Figure 1. But indeed, we are free to specify the functional dependence of the $\hat{\mathbf{1}}$-component on σ however we please. Being a contour path, the $\hat{\mathbf{3}}$-component must be constant with respect to σ (c.f. (7)):

$$\frac{\mathrm{d}}{\mathrm{d}\sigma}\,(h - (x - \sigma) - \mu) = 0 \quad \Rightarrow \quad \mu(\sigma) = y + \sigma$$

A z-contour path is therefore

$$\mathbf{c}(\sigma; z) \;=\; (x - \sigma)\hat{\mathbf{1}} + (y + \sigma)\,\hat{\mathbf{2}} + z\hat{\mathbf{3}} \tag{13}$$

And a corresponding tangent vector is

$$\frac{\mathrm{d}\mathbf{c}(\sigma; z)}{\mathrm{d}\sigma} \;=\; -\hat{\mathbf{1}} + \hat{\mathbf{2}} \tag{14}$$

Gradient path. For an $\mathbf{x}$-gradient path on $\mathcal{L}(h)$, we must calculate the partial derivaties

$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y} = -1$$

giving a tangent vector of $\mathbf{g}$, using (9), as

$$\frac{\mathrm{d}\mathbf{g}(\gamma; x, y)}{\mathrm{d}\gamma} \;=\; -\hat{\mathbf{1}} - \hat{\mathbf{2}} + 2\hat{\mathbf{3}} \tag{15}$$

From this tangent vector, it is easy to calculate an $\mathbf{x}$-gradient path on $\mathcal{L}(h)$ as

$$\mathbf{g}(\gamma; x, y) \;=\; (x-\gamma)\hat{\mathbf{1}} + (y-\gamma)\hat{\mathbf{2}} + (h-x-y+2\gamma)\hat{\mathbf{3}} \tag{16}$$

And as expected, the z-contour and $\mathbf{x}$-gradient paths intersect orthogonally at $\mathbf{x}$:

$$\begin{aligned}
&\mathbf{c}(0; z) = x\hat{\mathbf{1}} + y\hat{\mathbf{2}} + z\hat{\mathbf{3}} \\
&\mathbf{g}(0; x, y) = x\hat{\mathbf{1}} + y\hat{\mathbf{2}} + (h-x-y)\hat{\mathbf{3}} = x\hat{\mathbf{1}} + y\hat{\mathbf{2}} + z\hat{\mathbf{3}} \\
&\frac{\mathrm{d}\mathbf{c}(0; z)}{\mathrm{d}\sigma} \cdot \frac{\mathrm{d}\mathbf{g}(0; x, y)}{\mathrm{d}\gamma} = (-\hat{\mathbf{1}} + \hat{\mathbf{2}}) \cdot (-\hat{\mathbf{1}} - \hat{\mathbf{2}} + 2\hat{\mathbf{3}}) = 0
\end{aligned}$$

Natural coordinates. Using (10), the three normalised orthonormal basis vectors at position $\mathbf{x}$ on $\mathcal{L}(h)$ were calculated as:

$$\begin{aligned}
\hat{\mathbf{c}}(x, y) &= \frac{1}{\sqrt{2}}(-\hat{\mathbf{1}} + \hat{\mathbf{2}}) \\
\hat{\mathbf{g}}(x, y) &= \frac{1}{\sqrt{6}}(-\hat{\mathbf{1}} - \hat{\mathbf{2}} + 2\hat{\mathbf{3}}) \\
\hat{\mathbf{n}}(x, y) &= \frac{1}{\sqrt{3}}(\hat{\mathbf{1}} + \hat{\mathbf{2}} + \hat{\mathbf{3}})
\end{aligned}$$

To express $\mathbf{x}$ under $\{\hat{\mathbf{c}}, \hat{\mathbf{g}}, \hat{\mathbf{n}}\}$, write

$$\begin{aligned}
\mathbf{x}(x, y) &= x\hat{\mathbf{1}} + y\hat{\mathbf{2}} + z(x, y)\hat{\mathbf{3}} = x\hat{\mathbf{1}} + y\hat{\mathbf{2}} + (h-x-y)\hat{\mathbf{3}} \\
&= u(x, y)\hat{\mathbf{c}} + v(x, y)\hat{\mathbf{g}} + w(x, y)\hat{\mathbf{n}}
\end{aligned}$$

The resulting system of equations to be solved for u, v and w is

$$\begin{aligned}
-\sqrt{3}u - v + \sqrt{2}w &= \sqrt{6}x \\
\sqrt{3}u - v + \sqrt{2}w &= \sqrt{6}y \\
2v + \sqrt{2}w &= \sqrt{6}z
\end{aligned}$$

Solving this system for u, v and w gives

$$\mathbf{x}(x, y) = \frac{-x+y}{\sqrt{2}}\hat{\mathbf{c}}(x, y) + \frac{2h-3x-3y}{\sqrt{6}}\hat{\mathbf{g}}(x, y) + \frac{h}{\sqrt{3}}\hat{\mathbf{n}}(x, y)$$

Path traversal commutivity. On $\mathcal{L}(h)$, using (14), it is easy to calculate the length along the z-contour as

$$l(0, \sigma) = \int_0^\sigma \left|\frac{\mathrm{d}\mathbf{c}(\bar{\sigma}; z)}{\mathrm{d}\bar{\sigma}}\right| \mathrm{d}\bar{\sigma} = \sqrt{2}\sigma$$

This path length offers a natural parametrisation of the z-contour path on $\mathcal{L}(h)$. From (13),

$$\mathbf{c}(l; z) \;=\; \mathbf{c}(\sigma(l); z) \;=\; \left(x - l/\sqrt{2}\right)\hat{\mathbf{1}} \;+\; \left(y + l/\sqrt{2}\right)\hat{\mathbf{2}} \;+\; z\hat{\mathbf{3}}$$

On $\mathcal{L}(h)$, using (15), the length along the $\mathbf{x}$-gradient is

$$m(0, \gamma) = \int_0^\gamma \left|\frac{\mathrm{d}\mathbf{g}(\bar{\gamma}; x, y)}{\mathrm{d}\bar{\gamma}}\right| \mathrm{d}\bar{\gamma} = \sqrt{6}\,\gamma$$

and a natural parametrisation of the $\mathbf{x}$-gradient path on $\mathcal{L}(h)$ becomes, using (16),

$$\mathbf{g}(m; x, y) \;=\; \mathbf{g}(\gamma(m); x, y) \;=\; \left(x - m/\sqrt{6}\right)\hat{\mathbf{1}} \;+\; \left(y - m/\sqrt{6}\right)\hat{\mathbf{2}} \;+\; \left(h - x - y + 2m/\sqrt{6}\right)\hat{\mathbf{3}} \tag{17}$$

As per (12), define the coordinate mappings over $\mathcal{L}(h)$:

$$\begin{aligned}
\mathcal{C}(l) &: \mathbf{x} \mapsto \mathcal{C}(l)\mathbf{x} = \mathbf{c}(l; z) \\
\mathcal{G}(m) &: \mathbf{x} \mapsto \mathcal{G}(m)\mathbf{x} = \mathbf{g}(m; x, y)
\end{aligned}$$

A traversal of length l over the z-contour path, starting at position $\mathbf{x}$, followed by a traversal of length m over the $\mathbf{c}(l;z)$-gradient path is

$$\begin{aligned}\mathcal{G}(m)\mathcal{C}(l)\mathbf{x} &= \mathcal{G}(m)\left[\left(x - l/\sqrt{2}\right)\hat{\mathbf{1}} + \left(y + l/\sqrt{2}\right)\hat{\mathbf{2}} + z\hat{\mathbf{3}}\right] \\ &= \left[(x - l/\sqrt{2}) - m/\sqrt{6}\right]\hat{\mathbf{1}} + \left[(y + l/\sqrt{2}) - m/\sqrt{6}\right]\hat{\mathbf{2}} \\ &\quad + \left[h - (x - l/\sqrt{2}) - (y + l/\sqrt{2}) + 2m/\sqrt{6}\right]\hat{\mathbf{3}} \\ &= \left[x - l/\sqrt{2} - m/\sqrt{6}\right]\hat{\mathbf{1}} + \left[y + l/\sqrt{2} - m/\sqrt{6}\right]\hat{\mathbf{2}} + \left[h - x - y + 2m/\sqrt{6}\right]\hat{\mathbf{3}}\end{aligned}$$

And a traversal of length m over the $\mathbf{x}$-gradient path, starting at position $\mathbf{x}$, followed by a traversal of length l over the $\mathbf{g}_3(m;x,y)$-contour path is

$$\begin{aligned}\mathcal{C}(l)\mathcal{G}(m)\mathbf{x} &= \mathcal{C}(l)\left[\left(x - m/\sqrt{6}\right)\hat{\mathbf{1}} + \left(y - m/\sqrt{6}\right)\hat{\mathbf{2}} + \left(h - x - y + 2m/\sqrt{6}\right)\hat{\mathbf{3}}\right] \\ &= \left[(x - m/\sqrt{6}) - l/\sqrt{2}\right]\hat{\mathbf{1}} + \left[(y - m/\sqrt{6}) + l/\sqrt{2}\right]\hat{\mathbf{2}} + \left[h - x - y + 2m/\sqrt{6}\right]\hat{\mathbf{3}} \\ &= \mathcal{G}(m)\mathcal{C}(l)\mathbf{x}\end{aligned}$$

This proves the contour and gradient path traversal commutivity in the Euclidean flat plane. This commutivity underlies a meaningful definition of geometrically extended objects in Euclidean space, such a vectors. However, as will be shown in the next section, in non-Euclidean space this cannot be done, and the notion of a vector must be restricted to a geometrically local object, namely, the tangent vector.

7 Embedded inverted paraboloid

A typical inverted paraboloid $\mathcal{P}(h)$ (c.f. Eq. (3)) is shown in Figure 2.

Intuition. For a paraboloid, expressions for the contour and gradient paths are easy to obtain intuitively. If r and θ are the polar coordinates defined by the coordinate transformation

$$x(r,\theta) = r\cos\theta \quad \text{and} \quad y(r,\theta) = r\sin\theta$$

then contour and gradient paths passing through a specified point $(\rho\cos\phi, \rho\sin\phi)$ are, respectively,

$$\begin{aligned}\bar{\mathbf{c}}(\theta;\rho) &= \rho\cos\theta\,\hat{\mathbf{1}} + \rho\sin\theta\,\hat{\mathbf{2}} + \frac{1}{h}(h^2 - \rho^2)\hat{\mathbf{3}} \\ \bar{\mathbf{g}}(r;\phi) &= \cos\phi\, r\,\hat{\mathbf{1}} + \sin\phi\, r\,\hat{\mathbf{2}} + \frac{1}{h}(h^2 - r^2)\hat{\mathbf{3}}\end{aligned} \tag{18}$$

The notation "$\bar{\mathbf{c}}(\theta;\rho)$" indicates that the contour path is parametrised by the θ polar coordinate, with ρ a constant. Similarly, the notation "$\bar{\mathbf{g}}(r;\phi)$" indicates that the gradient path is parametrised by the r polar coodinate, with ϕ a constant. Clearly, as expected, $\bar{\mathbf{c}}(\phi;\rho) = \bar{\mathbf{g}}(\rho;\phi)$.

Respective tangent vectors at the point $(\rho\cos\phi, \rho\sin\phi)$ are:

$$\begin{aligned}\frac{\mathrm{d}\bar{\mathbf{c}}(\theta;\rho)}{\mathrm{d}\theta} &= -\rho\sin\theta\hat{\mathbf{1}} + \rho\cos\theta\hat{\mathbf{2}} \\ \frac{\mathrm{d}\bar{\mathbf{g}}(r;\phi)}{\mathrm{d}r} &= \cos\phi\hat{\mathbf{1}} + \sin\phi\hat{\mathbf{2}} - \frac{2r}{h}\hat{\mathbf{3}}\end{aligned}$$

And as expected, the two tangent vectors are orthogonal at the point $(\rho\cos\phi, \rho\sin\phi)$:

$$\begin{aligned}\frac{\mathrm{d}\bar{\mathbf{c}}(\theta;\rho)}{\mathrm{d}\theta} \cdot \frac{\mathrm{d}\bar{\mathbf{g}}(r;\phi)}{\mathrm{d}r} &= \rho\sin\phi\cos\theta - \rho\cos\phi\sin\theta \\ &= 0 \quad \text{whenever } \theta = \phi\end{aligned}$$

But now I wish not to appeal to basic intuition, but rather to the prescription laid out in (6), (7), (8) and (9).

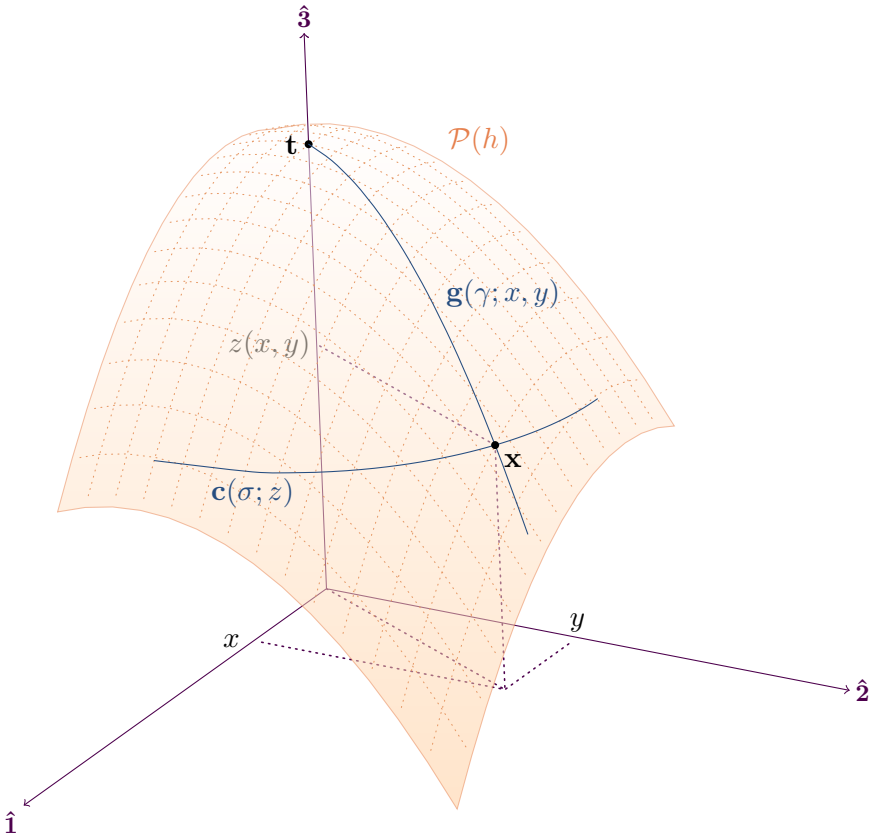


Figure 2: Inverted paraboloid $\mathcal{P}(h)$ specified by (3), showing a contour path $\mathbf{c}(\sigma; z)$ and a gradient path $\mathbf{g}(\gamma; x, y)$ intersecting orthogonally at the point $\mathbf{x}$ on $\mathcal{P}(h)$. Both $\mathbf{c}$ and $\mathbf{g}$ were calculated using (6), (7), (8) and (9), and drawn computationally. As expected, the apex position $\mathbf{t}$ lies on $\mathbf{g}$.

Contour path. Suppose that the point (x, y, z) lies on $\mathcal{P}(h)$. Then (x, y, z) must satisfy

$$z(x, y) = \frac{1}{h}\left[h^2 - x^2 - y^2\right]$$

$\mathcal{P}(h)$'s z-contour passing through the position $\mathbf{x} = x\hat{\mathbf{1}} + y\hat{\mathbf{2}} + z(x, y)\hat{\mathbf{3}}$ is the set $\{(x, y) \,|\, x^2 + y^2 = h(h - z)\}$. There is considerable latitude in specifying $\mathcal{P}(h)$'s z-contour path. We may choose that under a parametrisation, the path's $\hat{\mathbf{1}}$-component varies as $x - \sigma$. Then in place of (6), we may write for a z-contour path on $\mathcal{P}(h)$:

$$\mathbf{c}(\sigma; z) \;=\; (x - \sigma)\hat{\mathbf{1}} + \mu(\sigma)\hat{\mathbf{2}} + \frac{1}{h}\left[h^2 - (x - \sigma)^2 - \mu^2(\sigma)\right]\hat{\mathbf{3}} \quad \text{for some } \mu(\sigma).$$

In keeping with Figure 2, the $\hat{\mathbf{1}}$-component is parametrised as $x - \sigma$ so that the $\hat{\mathbf{1}}$-component decreases with increasing σ. Being a contour path, the $\hat{\mathbf{3}}$-component must be constant with respect to σ (c.f. (7)):

$$\begin{aligned}
&\frac{\mathrm{d}}{\mathrm{d}\sigma}\left(\frac{1}{h}\left[h^2 - (x - \sigma)^2 - \mu^2(\sigma)\right]\right) = 0 \\
\Rightarrow\quad & 2(x - \sigma) - 2\mu\frac{\mathrm{d}\mu}{\mathrm{d}\sigma} = 0 \\
\Rightarrow\quad & \int_y^{\mu(\sigma)} \mu \mathrm{d}\mu = \int_0^{\sigma} (x - \bar{\sigma})\mathrm{d}\bar{\sigma} \\
\Rightarrow\quad & \mu(\sigma) = \sqrt{x^2 + y^2 - (x - \sigma)^2} \quad \text{(Considering only the positive root for now.)}
\end{aligned}$$

This gives

$$\frac{1}{h}\left[h^2 - (x - \sigma)^2 - \mu^2(\sigma)\right] \;=\; \frac{1}{h}\left[h^2 - x^2 - y^2\right] \;=\; z$$

as expected. A z-contour path and its corresponding tangent vector is therefore

$$\begin{aligned}\mathbf{c}(\sigma;z) &= (x-\sigma)\hat{\mathbf{1}} + \sqrt{x^2+y^2-(x-\sigma)^2}\,\hat{\mathbf{2}} + z\hat{\mathbf{3}} \\ \frac{\mathrm{d}\mathbf{c}(\sigma;z)}{\mathrm{d}\sigma} &= -\hat{\mathbf{1}} + \frac{x-\sigma}{\sqrt{x^2+y^2-(x-\sigma)^2}}\hat{\mathbf{2}}\end{aligned} \tag{19}$$

To be sure, we could have chosen, say,

$$\mathbf{c}(u;z) = \sqrt{x^2+y^2-u^2}\,\hat{\mathbf{1}} \;+\; u\hat{\mathbf{2}} + z\hat{\mathbf{3}}$$

as an alternative but equally valid parametrisation.

Gradient path. For an $\mathbf{x}$-gradient path on $\mathcal{P}(h)$, we must calculate the partial derivatives

$$\frac{\partial z}{\partial x} = -\frac{2x}{h} \quad \text{and} \quad \frac{\partial z}{\partial y} = -\frac{2y}{h}$$

A tangent vector of $\mathbf{g}$ is, using (9),

$$\frac{\mathrm{d}\mathbf{g}(\gamma;x,y)}{\mathrm{d}\gamma} \;=\; -\frac{2g_1(\gamma)}{h}\hat{\mathbf{1}} - \frac{2g_2(\gamma)}{h}\hat{\mathbf{2}} + \frac{4}{h^2}\left[g_1^2(\gamma)+g_2^2(\gamma)\right]\hat{\mathbf{3}}$$

This can easily be solved for g_1 and g_2:

$$\begin{aligned}&\frac{\mathrm{d}g_1(\gamma)}{\mathrm{d}\gamma} = -\frac{2g_1(\gamma)}{h} \\ \Rightarrow\quad &\int_x^{g_1(\gamma)} \frac{\mathrm{d}g}{g} = -\frac{2}{h}\int_{\gamma_0}^{\gamma} \mathrm{d}\bar{\gamma} \\ \Rightarrow\quad &g_1(\gamma;x,y) = xe^{-2(\gamma-\gamma_0)/h}\end{aligned}$$

Similarly, $g_2(\gamma;x,y) = ye^{-2(\gamma-\gamma_0)/h}$. An $\mathbf{x}$-gradient path is therefore, from (8),

$$\mathbf{g}(\gamma;x,y) \;=\; xe^{-2(\gamma-\gamma_0)/h}\,\hat{\mathbf{1}} + ye^{-2(\gamma-\gamma_0)/h}\,\hat{\mathbf{2}} + \frac{1}{h}\left[h^2-(x^2+y^2)e^{-4(\gamma-\gamma_0)/h}\right]\hat{\mathbf{3}}$$

Provided that the parametrisation satisfies $\sigma_0 = 0$, it is clear that $\mathbf{c}(\sigma;z)$ and $\mathbf{g}(\gamma;x,y)$ intersect at the point on $\mathcal{P}(h)$ specified by (σ_0,γ_0), i.e., $\mathbf{c}(\sigma_0;z) = \mathbf{g}(\gamma_0;x,y)$. If we apply the coordinate transformation $e^{-2(\gamma-\gamma_0)/h} \mapsto \gamma$, with $\gamma_0 \to 1$, then the $\mathbf{x}$-gradient path and its corresponding tangent vector is

$$\begin{aligned}\mathbf{g}(\gamma;x,y) &= x\gamma\hat{\mathbf{1}} + y\gamma\hat{\mathbf{2}} + \frac{1}{h}\left[h^2-(x\gamma)^2-(y\gamma)^2\right]\hat{\mathbf{3}} \\ &= x\gamma\hat{\mathbf{1}} + y\gamma\hat{\mathbf{2}} + z(x\gamma,y\gamma)\hat{\mathbf{3}} \\ \frac{\mathrm{d}\mathbf{g}(\gamma;x,y)}{\mathrm{d}\gamma} &= x\hat{\mathbf{1}} + y\hat{\mathbf{2}} - \frac{2(x^2+y^2)\gamma}{h}\hat{\mathbf{3}}\end{aligned} \tag{20}$$

And at that point, their respective tangent vectors are orthogonal:

$$\frac{\mathrm{d}\mathbf{c}(0;z)}{\mathrm{d}\sigma} \cdot \frac{\mathrm{d}\mathbf{g}(1;x,y)}{\mathrm{d}\gamma} = \left(\hat{\mathbf{1}} - \frac{x}{y}\hat{\mathbf{2}}\right)\cdot\left(x\hat{\mathbf{1}} + y\hat{\mathbf{2}} - \frac{2(x^2+y^2)}{h}\hat{\mathbf{3}}\right) = 0$$

The prescription (6)…(9) was used to compute the paths (19) and (20), as shown in Figure 2. Refer to the Appendix on page 17 for details on the computations.

The prescription agrees with intuition. To prove correspondence between the two contour paths $\bar{\mathbf{c}}$ and $\mathbf{c}$, and between the two gradient paths $\bar{\mathbf{g}}$ and $\mathbf{g}$, it is sufficient to prove coincidence between respective positions on the pairs of paths. Comparing (18) and (19), a position on $\bar{\mathbf{c}}$ coincides with a position on $\mathbf{c}$ whenever

$$\rho\cos\theta = \sigma,\ \rho\sin\theta = \sqrt{x^2+y^2-\sigma^2},\ \text{and}\ \ \frac{1}{h}(h^2-\rho^2)$$

That is, whenever

$$\rho^2 = x^2+y^2$$

and

$$\tan\theta = \frac{\sqrt{x^2+y^2-\sigma^2}}{\sigma} \quad \Leftarrow \quad \theta = \arccos(\sigma/\sqrt{x^2+y^2})$$

That is, the following two contour paths correspond:

$$\begin{aligned}\bar{\mathbf{c}}(\theta;\sqrt{x^2+y^2}) &= \sqrt{x^2+y^2}\cos\theta\,\hat{\mathbf{1}} + \sqrt{x^2+y^2}\sin\theta\,\hat{\mathbf{2}} + z\hat{\mathbf{3}}\\ \mathbf{c}(\sigma;z) &= \sigma\hat{\mathbf{1}} + \sqrt{x^2+y^2-\sigma^2}\,\hat{\mathbf{2}} + z\hat{\mathbf{3}}\end{aligned}$$

And specifically,

$$\bar{\mathbf{c}}\left(\arccos(\sigma/\sqrt{x^2+y^2});\sqrt{x^2+y^2}\right) = \mathbf{c}(\sigma;z)$$

Comparing (18) and (20), a position on $\bar{\mathbf{g}}$ coincides with a position on $\mathbf{g}$ whenever

$$\cos\phi\, r = x\gamma \ \ \text{and} \ \ \sin\phi\, r = y\gamma,$$

That is, whenever

$$\phi = \arctan\left(\frac{y}{x}\right)$$

and

$$r = \sqrt{x^2+y^2}\,\gamma$$

We have thus proved that these two gradient paths correspond:

$$\begin{aligned}\bar{\mathbf{g}}(r;\arctan(y/x)) &= \frac{xr}{\sqrt{x^2+y^2}}\hat{\mathbf{1}} + \frac{yr}{\sqrt{x^2+y^2}}\hat{\mathbf{2}} + \frac{1}{h}\left(h^2-r^2\right)\hat{\mathbf{3}}\\ \mathbf{g}(\gamma;x,y) &= x\gamma\hat{\mathbf{1}} + y\gamma\hat{\mathbf{2}} + z(x\gamma,y\gamma)\hat{\mathbf{3}}\end{aligned}$$

And specifically,

$$\bar{\mathbf{g}}\left(\sqrt{x^2+y^2}\,\gamma;\arctan(y/x)\right) = \mathbf{g}(\gamma;x,y)$$

Natural coordinates. Using (19), (20), and evaluating the derivatives at $\sigma = 0$ and $\gamma = 1$, gives the orthonormal vector basis $\{\hat{\mathbf{c}},\hat{\mathbf{g}},\hat{\mathbf{n}}\}$ at position $\mathbf{x}$ on $\mathcal{P}(h)$:

$$\begin{aligned}\hat{\mathbf{c}}(x,y) &= \frac{1}{r}\left(-y\hat{\mathbf{1}} + x\hat{\mathbf{2}}\right)\\ \hat{\mathbf{g}}(x,y) &= \frac{1}{r\sqrt{h^2+4r^2}}\left(hx\hat{\mathbf{1}} + hy\hat{\mathbf{2}} - 2r^2\hat{\mathbf{3}}\right)\\ \hat{\mathbf{n}}(x,y) &= \frac{1}{\sqrt{h^2+4r^2}}\left(-2x\hat{\mathbf{1}} - 2y\hat{\mathbf{2}} - h\hat{\mathbf{3}}\right)\end{aligned}$$

with $r^2 = x^2+y^2$. To express $\mathbf{x}$ under $\{\hat{\mathbf{c}},\hat{\mathbf{g}},\hat{\mathbf{n}}\}$, write

$$\begin{aligned}\mathbf{x}(x,y) &= x\hat{\mathbf{1}} + y\hat{\mathbf{2}} + z(x,y)\hat{\mathbf{3}}\\ &= x\hat{\mathbf{1}} + y\hat{\mathbf{2}} + \frac{1}{h}\left(h^2-x^2-y^2\right)\hat{\mathbf{3}}\\ &= u(x,y)\hat{\mathbf{c}} + v(x,y)\hat{\mathbf{g}} + w(x,y)\hat{\mathbf{n}}\end{aligned}$$

The resulting system of equations to be solved for u, v and w is

$$\begin{aligned} -y\sqrt{h^2+4r^2}\,u + hxv - 2xrw &= xr\sqrt{h^2+4r^2} \\ x\sqrt{h^2+4r^2}\,u + hyv - 2yrw &= yr\sqrt{h^2+4r^2} \\ -2rv - hw &= \sqrt{h^2+4r^2}z \end{aligned}$$

This system is linear in u, v and w, and is easily solved, giving

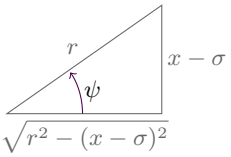
$$\mathbf{x}(x,y) = \frac{1}{\sqrt{h^2+4(x^2+y^2)}}\left(0\hat{\mathbf{c}}(x,y) \;-\; \frac{\sqrt{x^2+y^2}}{h}\left(h^2-2(x^2+y^2)\right)\hat{\mathbf{g}}(x,y) \;-\; (h^2+x^2+y^2)\hat{\mathbf{n}}(x,y)\right)$$

Path traversal commutivity. On $\mathcal{P}(h)$, using (11) and (19), the length along the z-contour is

$$l(0,\sigma) = \int_0^\sigma \left|\frac{\mathrm{d}\mathbf{c}(\bar{\sigma};z)}{\mathrm{d}\bar{\sigma}}\right| \mathrm{d}\bar{\sigma} = \int_0^\sigma \frac{r}{\sqrt{r^2-(x-\bar{\sigma})^2}}\mathrm{d}\bar{\sigma} \quad \text{with } r=\sqrt{x^2+y^2}$$

Although this is a well-known standard integral,[1] I wish to carry out the integration here. Let $x-\bar{\sigma} = r\arcsin\bar{\psi}$. Then $\mathrm{d}\bar{\sigma} = -r\cos\bar{\psi}\mathrm{d}\bar{\psi}$, giving

$$\begin{aligned} l(0,\sigma) &= \int_{\psi_0}^{\psi} \frac{-r^2\cos\bar{\psi}}{r\sqrt{1-\sin^2\bar{\psi}}}\,\mathrm{d}\bar{\psi} \quad \text{with } \psi_0 = \arcsin\left(\frac{x}{r}\right),\ \psi = \arcsin\left(\frac{x-\sigma}{r}\right) \\ &= r(\psi_0-\psi) \\ &= r\arcsin\left(\sin(\psi_0-\psi)\right) \\ &= r\arcsin\left(\sin\psi_0\cos\psi - \sin\psi\cos\psi_0\right) \\ &= r\arcsin\left(\frac{x}{r}\frac{\sqrt{r^2-(x-\sigma)^2}}{r} - \frac{x-\sigma}{r}\frac{\sqrt{r^2-x^2}}{r}\right) \\ &= r\arcsin\left(\frac{x\sqrt{r^2-(x-\sigma)^2}-y(x-\sigma)}{r^2}\right) \end{aligned} \tag{21}$$



Consider traversing $\mathcal{P}(h)$'s 0-contour path (c.f. (19)) $\mathbf{c}(\sigma;0) = (h-\sigma)\hat{\mathbf{1}} + \sqrt{h^2-(h-\sigma)^2}\,\hat{\mathbf{2}} + 0\hat{\mathbf{3}}$, beginning at $\mathbf{c}(0;0)$ and ending at $\mathbf{c}(h;0)$. Then
$l(0,h) = h\arcsin((h\sqrt{h^2-0}-0)/h^2) = h\arcsin(1) = \pi h/2$, as expected.

Equation (21) provides length traversed along $\mathbf{c}(\sigma;z)$ as a function of the path parameter σ, starting at the position $\mathbf{x}$ on $\mathcal{P}(h)$. This relationship can easily be inverted, allowing σ to be specified as a function of distance traversed:

$$\sigma(l) = x + y\sin(l/r) - x\cos(l/r) \tag{22}$$

Substitution into (19) offers a natural parametrisation of the z-contour path on $\mathcal{P}(h)$ as

$$\mathbf{c}(l;z) = \left[x\cos(l/r) - y\sin(l/r)\right]\hat{\mathbf{1}} \;+\; \left[x\sin(l/r) + y\cos(l/r)\right]\hat{\mathbf{2}} \;+\; z\hat{\mathbf{3}} \tag{23}$$

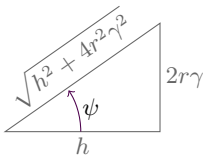
Using (11) and (20), the length along the $\mathbf{x}$-gradient is

$$m(1,\gamma) = \int_1^\gamma \left|\frac{\mathrm{d}\mathbf{g}(\bar{\gamma};x,y)}{\mathrm{d}\bar{\gamma}}\right| \mathrm{d}\bar{\gamma} = \int_1^\gamma r\sqrt{1+\frac{4r^2\bar{\gamma}^2}{h^2}}\,\mathrm{d}\bar{\gamma} \quad \text{with } r=\sqrt{x^2+y^2}$$

Again, although this is a well-known integral,[1] I choose to carry out the integration here. Applying

the substitutions $\bar{\gamma} = \dfrac{h}{2r}\tan\bar{\psi}$, $\mathrm{d}\bar{\gamma} = \dfrac{h}{2r}\sec^2\bar{\psi}\,\mathrm{d}\bar{\psi}$, gives

$$\begin{aligned}
\frac{2}{h}\,m(1,\gamma) &= \int_{\psi_0}^{\psi} \sec^3\bar{\psi}\,\mathrm{d}\bar{\psi} \quad \text{with } \psi = \arctan\left(\frac{2r\gamma}{h}\right),\ \psi_0 = \arctan\left(\frac{2r}{h}\right) \\
&= \int_{\psi_0}^{\psi} \sec\bar{\psi}\sec^2\bar{\psi}\,\mathrm{d}\bar{\psi} \\
&= \int_{\psi_0}^{\psi} \sec\bar{\psi}\,\frac{\mathrm{d}\tan\bar{\psi}}{\mathrm{d}\bar{\psi}}\,\mathrm{d}\bar{\psi} \\
&= \sec\bar{\psi}\tan\bar{\psi}\Big|_{\psi_0}^{\psi} - \int_{\psi_0}^{\psi}\frac{\mathrm{d}\sec\bar{\psi}}{\mathrm{d}\bar{\psi}}\tan\bar{\psi}\,\mathrm{d}\bar{\psi} \quad \text{(Using "Integration by parts".)} \\
&= \sec\bar{\psi}\tan\bar{\psi}\Big|_{\psi_0}^{\psi} - \int_{\psi_0}^{\psi}\sec\bar{\psi}\tan^2\bar{\psi}\,\mathrm{d}\bar{\psi} \\
&= \sec\bar{\psi}\tan\bar{\psi}\Big|_{\psi_0}^{\psi} - \int_{\psi_0}^{\psi}\sec\bar{\psi}\left(\sec^2\bar{\psi}-1\right)\mathrm{d}\bar{\psi} \\
&= \sec\bar{\psi}\tan\bar{\psi}\Big|_{\psi_0}^{\psi} - \frac{2}{h}\,m(1,\gamma) + \int_{\psi_0}^{\psi}\sec\bar{\psi}\,\mathrm{d}\bar{\psi} \\
\Rightarrow\quad \frac{4}{h}\,m(1,\gamma) &= \sec\bar{\psi}\tan\bar{\psi}\Big|_{\psi_0}^{\psi} + \ln\left(\sec\bar{\psi}+\tan\bar{\psi}\right)\Big|_{\psi_0}^{\psi} \\
&= \sec\psi\tan\psi - \sec\psi_0\tan\psi_0 + \ln\left(\frac{\sec\psi+\tan\psi}{\sec\psi_0+\tan\psi_0}\right) \\
&= \frac{2r}{h^2}\left(\gamma\sqrt{h^2+4r^2\gamma^2} - \sqrt{h^2+4r^2}\right) + \ln\left(\frac{2r\gamma+\sqrt{h^2+4r^2\gamma^2}}{2r+\sqrt{h^2+4r^2}}\right) \\
&= \left(\frac{2r\gamma}{h}\right)\sqrt{1+\left(\frac{2r\gamma}{h}\right)^2} - \left(\frac{2r}{h}\right)\sqrt{1+\left(\frac{2r}{h}\right)^2} + \\
&\qquad \ln\left(\left(\frac{2r\gamma}{h}\right)-\sqrt{1+\left(\frac{2r\gamma}{h}\right)^2}\right) - \ln\left(\left(\frac{2r}{h}\right)-\sqrt{1+\left(\frac{2r}{h}\right)^2}\right) \\
\Rightarrow\quad m(1,\gamma) &= \frac{r}{2h}\left(\gamma\sqrt{h^2+4r^2\gamma^2} - \sqrt{h^2+4r^2}\right) + \frac{h}{4}\ln\left(\frac{2r\gamma+\sqrt{h^2+4r^2\gamma^2}}{2r+\sqrt{h^2+4r^2}}\right)
\end{aligned}$$



But if we define $\beta(u) = 2\,\mathrm{arcsinh}(2\sqrt{x^2+y^2}\,u/h)$, and keep in mind these hyperbolic function identities:[1]

$$\begin{aligned}
&\mathrm{arcsinh}(x) = \ln(x+\sqrt{x^2+1}) \\
&\cosh^2(x) - \sinh^2(x) = 1 \\
&\sinh(x)\cosh(x) = \frac{1}{2}\sinh(2x) \\
&\sinh(x) - \sinh(y) = 2\cosh(\frac{x+y}{2})\sinh(\frac{x-y}{2})
\end{aligned}$$

then

$$\begin{aligned}
\frac{4}{h}\,m(1,\gamma) &= \sinh\left(\frac{\beta(\gamma)}{2}\right)\sqrt{1+\sinh^2\left(\frac{\beta(\gamma)}{2}\right)} - \sinh\left(\frac{\beta(1)}{2}\right)\sqrt{1+\sinh^2\left(\frac{\beta(1)}{2}\right)} \\
&\quad + \frac{\beta(\gamma)-\beta(1)}{2} \\
&= \sinh\left(\frac{\beta(\gamma)}{2}\right)\cosh\left(\frac{\beta(\gamma)}{2}\right) - \sinh\left(\frac{\beta(1)}{2}\right)\cosh\left(\frac{\beta(1)}{2}\right) \\
&\quad + \frac{\beta(\gamma)-\beta(1)}{2}
\end{aligned}$$

so that finally,

$$m(1,\gamma) = \frac{h}{8}\left(\sinh\beta(\gamma) + \beta(\gamma) - \sinh\beta(1) - \beta(1)\right) \quad \text{with } \beta(u) = 2\,\mathrm{arcsinh}\left(2\sqrt{x^2+y^2}\,u/h\right)$$

This expression for $m(1,\gamma)$ cannot be inverted analytically, as was done for $l(0,\sigma)$ in (22). So unfortunately, we are unable to obtain an analytical expression for the natural parametrisation of the $\mathbf{x}$-gradient path on $\mathcal{P}(h)$, as was done for $\mathcal{L}(h)$ in (17). But observe that $m(1,\gamma)$'s dependence on x and y is only in the form of powers of $r = \sqrt{x^2+y^2}$. And any point on the z-contour path on $\mathcal{P}(h)$ preserves $\sqrt{x^2+y^2}$ (Eq. (19)). Therefore, the parametrisations are the same for the path length of all $\mathbf{x}$-gradients such that $\mathbf{x}$ lies on the z-contour path. And so, to analyse path traversal commutivity of the z-contour path and the $\mathbf{x}$-gradient for any $\mathbf{x}$ on $\mathcal{P}(h)$, we may happily use $\mathbf{c}(l;z)$ from (23) together with $\mathbf{g}(\gamma;x,y)$ from (20).

As suggested by (12), and in keeping with the previous paragraph, define the coordinate mappings over $\mathcal{P}(h)$:

$$\mathcal{C}(l) : \mathbf{x} \mapsto \mathcal{C}(l)\mathbf{x} = \mathbf{c}(l;z)$$
$$\mathcal{G}(\gamma) : \mathbf{x} \mapsto \mathcal{G}(\gamma)\mathbf{x} = \mathbf{g}(\gamma;x,y)$$

A traversal of length l over the z-contour path, starting at position $\mathbf{x}$, followed by a traversal of length $m(1,\gamma)$ over the $\mathbf{c}(l;z)$-gradient path is

$$\begin{aligned}\mathcal{G}(\gamma)\mathcal{C}(l)\mathbf{x} &= \mathcal{G}(\gamma)\left[\,(x\cos(l/r) - y\sin(l/r))\,\hat{\mathbf{1}} + (x\sin(l/r) + y\cos(l/r))\,\hat{\mathbf{2}} + z\hat{\mathbf{3}}\,\right] \\ &= \left[x\cos(l/r) - y\sin(l/r)\right]\gamma\hat{\mathbf{1}} + \left[x\sin(l/r) + y\cos(l/r)\right]\gamma\hat{\mathbf{2}} + \frac{1}{h}\left[h^2 - r^2\gamma^2\right]\hat{\mathbf{3}}\end{aligned}$$

And a traversal of length $m(1,\gamma)$ over the $\mathbf{x}$-gradient path, starting at position $\mathbf{x}$, followed by a traversal of length l over the $\mathbf{g}_3(m(1,\gamma);x,y)$-contour path is

$$\begin{aligned}\mathcal{C}(l)\mathcal{G}(\gamma)\mathbf{x} \;&=\; \mathcal{C}(l)\left[x\gamma\hat{\mathbf{1}} + y\gamma\hat{\mathbf{2}} + \frac{1}{h}\left(h^2 - r^2\gamma^2\right)\hat{\mathbf{3}}\right] \\ &=\; \left[x\gamma\cos\left(l/\sqrt{(x\gamma)^2+(y\gamma)^2}\right) - y\gamma\sin\left(l/\sqrt{(x\gamma)^2+(y\gamma)^2}\right)\right]\hat{\mathbf{1}} + \\ &\quad \left[x\gamma\sin\left(l/\sqrt{(x\gamma)^2+(y\gamma)^2}\right) + y\gamma\cos\left(l/\sqrt{(x\gamma)^2+(y\gamma)^2}\right)\right]\hat{\mathbf{2}} + \\ &\quad \frac{1}{h}\left[h^2 - (x\gamma) - (y\gamma)^2\right]\hat{\mathbf{3}} \\ &=\; \left[x\cos\left(l/r\gamma\right) - y\sin\left(l/r\gamma\right)\right]\gamma\hat{\mathbf{1}} + \\ &\quad \left[x\sin\left(l/r\gamma\right) + y\cos\left(l/r\gamma\right)\right]\gamma\hat{\mathbf{2}} + \\ &\quad \frac{1}{h}\left[h^2 - r^2\gamma^2\right]\hat{\mathbf{3}}\end{aligned}$$

Taking the difference:

$$\begin{aligned}(\mathcal{G}(\gamma)\mathcal{C}(l) - \mathcal{C}(l)\mathcal{G}(\gamma))\mathbf{x} \;&=\; \gamma\left[x\cos(l/r) - y\sin(l/r) - x\cos(l/r\gamma) + y\sin(l/r\gamma)\right]\hat{\mathbf{1}} + \\ &\quad \gamma\left[x\sin(l/r) + y\cos(l/r) - x\sin(l/r\gamma) - y\cos(l/r\gamma)\right]\hat{\mathbf{2}} \\ &=\; \gamma\left[x\left[\cos(l/r) - \cos(l/r\gamma)\right] - y\left[\sin(l/r) - \sin(l/r\gamma)\right]\right]\hat{\mathbf{1}} + \\ &\quad \gamma\left[x\left[\sin(l/r) - \sin(l/r\gamma)\right] + y\left[\cos(l/r) - \cos(l/r\gamma)\right]\right]\hat{\mathbf{2}}\end{aligned}$$

The trigonometric identities

$$\cos(a) - \cos(b) = 2\sin(\frac{a+b}{2})\sin(\frac{b-a}{2})$$
$$\sin(a) - \sin(b) = 2\cos(\frac{a+b}{2})\sin(\frac{a-b}{2})$$

give, after some algebraic manipulation,

$$\begin{aligned}&(\mathcal{G}(\gamma)\mathcal{C}(l) - \mathcal{C}(l)\mathcal{G}(\gamma))\,\mathbf{x} \\ &\quad= 2\gamma\sin\left(\frac{l}{2r}\left(1-\frac{1}{\gamma}\right)\right)\left\{\left[-x\sin\left(\frac{l}{2r}\left(1+\frac{1}{\gamma}\right)\right) - y\cos\left(\frac{l}{2r}\left(1+\frac{1}{\gamma}\right)\right)\right]\hat{\mathbf{1}}\right. \\ &\qquad\left. + \left[x\cos\left(\frac{l}{2r}\left(1+\frac{1}{\gamma}\right)\right) - y\sin\left(\frac{l}{2r}\left(1+\frac{1}{\gamma}\right)\right)\right]\hat{\mathbf{2}}\right\}\end{aligned} \tag{24}$$

Clearly, this path traversal difference vector vanishes only when $\gamma = 1$ or when $l = 0$. The two path traversal pairs over $\mathcal{P}(h)$ shown in Figure 3 demonstrate the non-vanishing of (24).

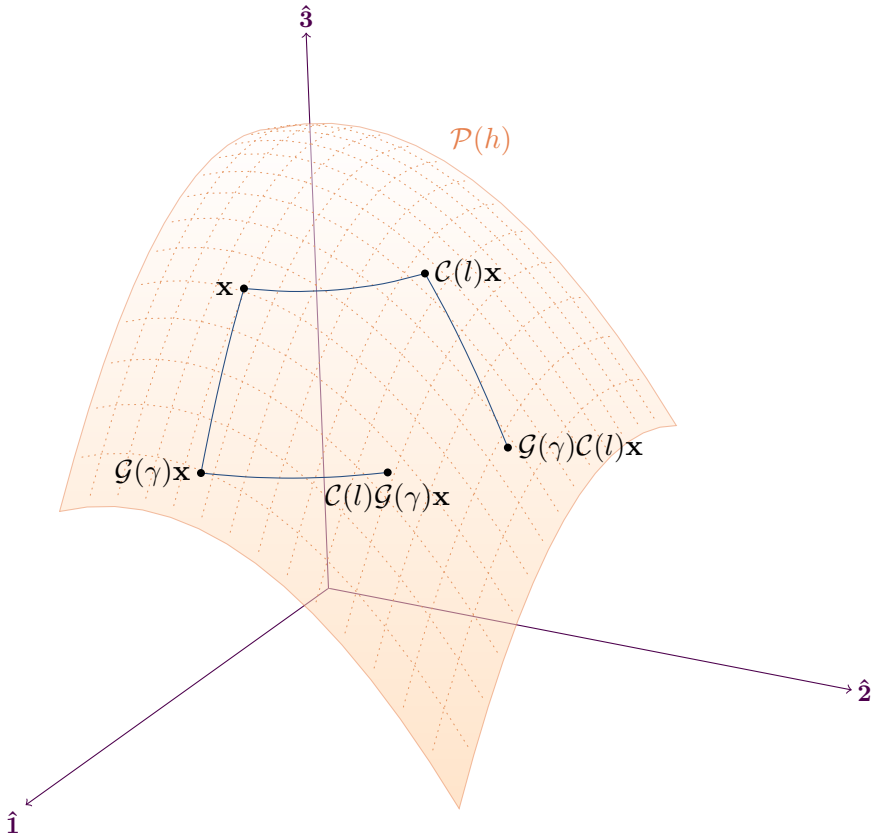


Figure 3: On the inverted paraboloid $\mathcal{P}(h)$, contour path and gradient path traversals, as calculated in (24), do not commute. The four paths were calculated using (6), (7), (8) and (9), and drawn computationally. Refer to the appendix in Section 9 for details on the computations.

8 Acknowledgments

As always Mels, thanks for being such a close friend and supportive partner, and for showing an interest in this work.

9 Appendix—Computed drawing with LaTeX, Ti*k*Z, pkTi*k*Z and PKREALVECTOR

In this section I demonstrate the combined use of LaTeX, Ti*k*Z, my pkTi*k*Z LaTeX package[?], and my C object class called PKREALVECTOR[2] to produce the three-dimensional schematic diagrams included in this document.

Perhaps not suprisingly, the text in the document was typeset with LaTeX. The figures were typeset with Ti*k*Z. Ti*k*Z is software capability for typesetting graphical content directly in LaTeX. The specification and calculation of the three-dimensional landscapes in the figures were done in the C programming language with the help of my PKREALVECTOR object class. PKREALVECTOR provides a useful coding abstraction for instantiating and manipulating vectors in $\mathbb{R}^n$. For example, PKREALVECTOR's API[2] includes calls to perform the rotational coordinate transformations needed to render on paper a two-dimensional projection of a three-dimensional landscape.

To typeset a figure, the LaTeX source file for this document `\input{}`'s another external LaTeX source file. In the case of Figure 2, the file was named `paraboloidfigure.tex`. The file contains the Ti*k*Z source code instructions to typeset the figure. The file was generated dynamically as the output of the execution of the `paraboloidfigure.run` program, which in turn was created by compiling the C code located in the `paraboloidfigure.c` file. Actually, by virtue of the presence of the `Makefile` file for the UNIX Make system, as listed below, I simply needed to type `make` to create the final PDF-formatted document file, a copy of which you are currently reading.

To incorporate Ti*k*Z's capabilities during typesetting, I included the following lines of LaTeX code in the preamble of my LaTeX "`.tex`" file:

```
\usepackage{pktikz}
\usetikzlibrary{calc}
\usetikzlibrary{positioning}
\usetikzlibrary{intersections}
```

Ti*k*Z was customised "globally" for all figures in the document using the following lines of LaTeX code:

```
1  \newcommand*\st{\,|\,} % 'such that'.
2  \newcommand*\ud{\text d}
3  \newcommand*\deriv[2]{\frac{\ud #1}{\ud #2}}
4  \newcommand*\derivB[2]{\ud #1/\ud #2}
5  \newcommand*\parDeriv[2]{\frac{\partial #1}{\partial #2}}
6  \newcommand*\evalAt[2]{\left.#1\right|_{#2}}%
7  \newcommand*\evalFromTo[3]{\left.#1\right|_{#2}^{#3}}%
8
9  \newcommand*\abs[1]{\protect\left|#1\protect\right|}
10
11 \newcommand*\one{\pktikzBasisVector{1}}
12 \newcommand*\two{\pktikzBasisVector{2}}
13 \newcommand*\three{\pktikzBasisVector{3}}
14
15 \newcommand*\vecx{\pktikzVector{x}}
16 \newcommand*\vecc{\pktikzVector{c}}
17 \newcommand*\vecg{\pktikzVector{g}}
18 \newcommand*\vecG{\pktikzVector{G}}
19 \newcommand*\vect{\pktikzVector{t}}
20 \newcommand*\surface{\mathcal{S}}
21 \newcommand*\plane{\mathcal{L}(h)}
22 \newcommand*\parab{\mathcal{P}(h)}
```

[2] Application Programming Interface

```
23  \newcommand*\chat{\pktikzUnitVector{c}}
24  \newcommand*\ghat{\pktikzUnitVector{g}}
25  \newcommand*\nhat{\pktikzUnitVector{n}}
26  \newcommand*\contmap{\mathcal{C}}
27  \newcommand*\gradmap{\mathcal{G}}
28  \newcommand*\dS{\ud S}
29  \newcommand*\vecdS{\ud\pktikzVector{S}}
30
31  \newcommand*\myrsqr{x^2+y^2}
32  \newcommand*\myr{\sqrt{\myrsqr}}
33
34  %-------------------------------------------------------------
35  % For TikZ begins.
36  %
37  %\tikzset{
38  %}
39  %
40  % For TikZ ends.
41  %-------------------------------------------------------------
```

The file `paraboloidfigure.tex` contains Ti*k*Z code for Figure 2. It resembles:

```
\begin{PkTikzpicture}[scale=1.0]
   %
   % Some coordinates.
   %
   \coordinate (origin) at (0,0);
   %
   % Basisvectors.
   %
   \draw[pktikzbasisvector,<->]
      (-4.00405,-2.86636) node[below left] {$\one$} --
      (origin) -- (6.91956,-1.34503) node[right] {$\two$};
   .
   .
   .
   %
   % The paraboloid.
   %
   \draw[parabsurface]
      (-3.55594,1.01518) --
      (-3.19215,1.07405) --
      . . .
      (-3.55594,1.01518);
   \path (1.53038,5.65063) node[above right,edgecolor]{$\parab$};
   \draw[surface]
      (-0.90268,6.05877) --
      (-0.538888,6.11765) --
      . . .
      (4.38155,2.12884);
   %
   % 'z'-contour path 'c'.
   %
   \draw[emphvectorcolor] plot[smooth] coordinates {
      (3.58346,2.51076)
      (3.46522,2.4308)
      . . .
```

```
      (-2.27789,1.69415) };
   \path (-0.976833,1.5442) node[below,emphvectorcolor]{$\vecc(\sigma;z)$};
   .
   .
   .
\end{PkTikzpicture}
```

The `paraboloidfigure.tex` file was incorporated into the body of the text with an `\input{}` LaTeX command, as follows:

```
\begin{figure}[h!]
   \begin{center}
      \input{paraboloidfigure.tex}
   \end{center}
   \caption{...}
   \label{paraboloid}
\end{figure}
```

The content of the `paraboloidfigure.c` C source file, which was used to create the Ti*k*Z code in `paraboloidfigure.tex`, has been primed to be typeset using the pkTechDoc "literate programming" LaTeX package.[3] pkTechDoc makes it possible to closely juxtapose LaTeX code and non-LaTeX code both for typesetting and for compilation outside of LaTeX.

```
1   #include <pkfeatures.h>
2
3   #include <stddef.h>
4   #include <stdlib.h>
5   #include <stdio.h>
6   #include <unistd.h>
7   #include <stdarg.h>
8   #include <string.h>
9   #include <math.h>
10  #include <float.h>
11
12  #include <pkmemdebug.h>
13  #include <pkerror.h>
14  #include <pktypes.h>
15  #include <pkstring.h>
16  #include <pkmath.h>
17  #include <pkrealvector.h>
18
19  #include "globals.h"
20
21  /*-------------------------------------------------------------*/
22
23  const char *LOGFNAME = "/tmp/diagram.log";
24
25  /*-------------------------------------------------------------*/
26
27  static void _printVector( const PKREALVECTOR *v )
28  {
29     printf( "Vector %s = ( ", pkRealVectorGetName(v) );
30     pkRealVectorPrintf( v, "__COMPONENTVALUE__", ", " );
31     puts(" )");
32     return;
33  }
```

The `_paraboloid()` private function below simply returns the value $z(x, y)$ of the constitutive equation (3) for the inverted paraboloid $\mathcal{P}(h)$. The `paraHeight` global variable is defined in the `globals.h` C header file.

```
34  static PKMATHREAL _paraboloid( const PKMATHREAL x, const PKMATHREAL y )
35  {
36     const PKMATHREAL h = paraHeight;
37     return( 1.0 / h * ( h*h - x*x - y*y ) );
38  }
```

The `_diagram()` private function below specifies the diagram's three-dimensional landscape. It does this primarily using the `PKREALVECTOR` object class. The function prints to standard output a body of Ti*k*Z source code which may be used to typeset the landscape in LaTeX.

But before this function can do so, it must transform the landscape in such a way that what Ti*k*Z typesets is a two-dimensional projection of the three-dimensional landscape. The function rotationally transforms the landscape onto the space spanned by the $\{\hat{\mathbf{1}}', \hat{\mathbf{2}}', \hat{\mathbf{3}}'\}$ orthonormal vector basis set, where the $\hat{\mathbf{1}}'$ and $\hat{\mathbf{2}}'$ basis vectors lie in the plane of the page and $\hat{\mathbf{3}}'$ is perpendicular to the page, i.e., lying parallel to the reader's line of sight.

In the function, the `xLos`, `yLos` and `zLos` are required for the rotational transformations. They are the three coordinates of the "line-of-sight" vector under the $\{\hat{\mathbf{1}}, \hat{\mathbf{2}}, \hat{\mathbf{3}}\}$ vector basis. The angle θ is the tilt angle between $\hat{\mathbf{2}}$ and the $\hat{\mathbf{2}}'\hat{\mathbf{3}}'$ plane. The actual transformation is affected via calls resembling

```
pkRealVectorsUnderLineOfSightBasis1( xLos, yLos, zLos, theta,
                                     e1, e2, e3,
                                     ...,
                                     NULL )
```

For further details, refer to [2] and [4].

```
39  static void _diagram(void)
40  {
41     const int Nx = 15,
42               Ny = Nx,
43               Nc = 20,
44               Ng = 10;
45     PKMATHREAL xMin,
46                xMax,
47                yMin,
48                yMax;
49     PKMATHREAL p, q;
50     PKREALVECTOR *e1,
51                  *e2,
52                  *e3,
53                  *t,     /* Apex of the paraboloid. */
54                  *x,     /* Position on the paraboloid. */
55                  *x1,    /* Component vector of 'x' along 1. */
56                  *x2,    /* Component vector of 'x' along 2. */
57                  *x3,    /* Component vector of 'x' along 3. */
58                  *x12,   /* Component vector of 'x' in the 1-2 plane. */
59                  *r[Nx+1][Ny+1], /* An array of positions on the paraboloid. */
60                  *c[Nc+1],       /* An array of positions on the contour path */
61                                  /* passing thru 'x'. */
62                  *g[Ng+1];       /* An array of positions on the gradient path */
63                                  /* passing thru 'x'. */
64        int i,
65            j;
```

Here we specify the three-dimensional landscape. Begin with the $\{\hat{\mathbf{1}}, \hat{\mathbf{2}}, \hat{\mathbf{3}}\}$ vector basis.

```
66      e1 = pkRealVectorAlloc1( "\\one",   3, basisVecLen, 0.0, 0.0 );
67      e2 = pkRealVectorAlloc1( "\\two",   3, 0.0, basisVecLen, 0.0 );
68      e3 = pkRealVectorAlloc1( "\\three", 3, 0.0, 0.0, basisVecLen );
69
70      /*
71       * The apex position.
72       */
73      t = pkRealVectorAlloc1( "\\vect", 3, 0.0, 0.0, _paraboloid(0.0,0.0) );
```

The array $\mathbf{r}$ represents an $N_x \times N_y$ matrix of positions vectors on $\mathcal{P}(h)$.

```
74      {
75         xMin = - 0.2 * paraHalfWidth;
76         xMax =   0.8 * paraHalfWidth;
77         yMin = - 0.2 * paraHalfWidth;
78         yMax =   0.8 * paraHalfWidth;
79         for ( i = 0;  i < Nx;  i++ ) {
80            p = xMin + (double)i / (double)Nx * ( xMax - xMin );
81            for ( j = 0;  j < Ny;  j++ ) {
82               q = yMin + (double)j / (double)Ny * ( yMax - yMin );
83               r[i][j] = pkRealVectorAlloc1( "", 3, p, q, _paraboloid(p,q) );
84            }
85         }
86      }
```

The position vector $\mathbf{x}$ represent any point of interest on $\mathcal{P}(h)$.

```
87      p = 0.3 * paraHeight;
88      q = 0.6 * paraHeight;
89      x = pkRealVectorAlloc1( "\\vecx", 3, p, q, _paraboloid(p,q) );
90      x12 = pkRealVectorAlloc1( "x12", 3, p, q, 0.0 );
91      x1 = pkRealVectorAlloc1( "x", 3, p, 0.0, 0.0 );
92      x2 = pkRealVectorAlloc1( "y", 3, 0.0, q, 0.0 );
93      x3 = pkRealVectorAlloc1( "z(x,y)", 3, 0.0, 0.0, _paraboloid(p,q) );
```

Specification of N_c sample positions on the z-contour path $\mathbf{c}(\sigma; z)$.

```
94      {
95         const PKMATHREAL x0 = pkRealVectorGetComponent(x)[0],
96                          y0 = pkRealVectorGetComponent(x)[1],
97                          z0 = _paraboloid(x0,y0),
98         //xMin = - sqrt( x0 * x0 + y0 * y0 ) + FLT_EPSILON;
99         xMin = 0.0;
100        xMax =   sqrt( x0 * x0 + y0 * y0 ) - FLT_EPSILON;
101        for ( i = 0;  i < Nc;  i++ ) {
102           char *name = strAllocPrintf( "\\vecc_%d", i );
103           p = xMin + (double)i / (double)(Nc - 1) * ( xMax - xMin );
104           q = sqrt( y0 * y0 + x0 * x0 - p * p );
105           c[i] = pkRealVectorAlloc1( name, 3, p, q, z0 );
106           strFreePrintf(name);
107        }
108     }
```

Specification of N_g sample positions on the $\mathbf{x}$-contour path $\mathbf{g}(\gamma; x, y)$.

```
109     {
110        const PKMATHREAL x0 = pkRealVectorGetComponent(x)[0],
111                         y0 = pkRealVectorGetComponent(x)[1];
```

```
112         xMin = pkRealVectorGetComponent(x)[0] + 0.05 * paraHalfWidth;
113         xMax = pkRealVectorGetComponent(t)[0];
114         for ( i = 0;  i < Ng;  i++ ) {
115            char *name = strAllocPrintf( "\\vecg_%d", i );
116            p = xMin + (double)i / (double)(Ng - 1) * ( xMax - xMin );
117            q = y0 / x0 * p;
118            g[i] = pkRealVectorAlloc1( name,
119                                       3,
120                                       p,
121                                       q,
122                                       _paraboloid(p,q) );
123            strFreePrintf(name);
124         }
125      }
```

Rotationally transform the landscape onto the space spanned by the abovementioned "line-of-site" basis. That is, transform all vectors into "shadow" vectors which lie in the $\hat{\mathbf{1}}'\hat{\mathbf{2}}'$ plane lying flat on the page.

```
126      for ( i = 0;  i < Nx;  i++ ) {
127         for ( j = 0;  j < Ny;  j++ ) {
128            pkRealVectorUnderLineOfSightBasis1( r[i][j],
129                                                xLos, yLos, zLos, theta );
130         }
131      }
132      pkRealVectorsUnderLineOfSightBasisV1( xLos, yLos, zLos, theta,
133                                            c, Nc );
134      pkRealVectorsUnderLineOfSightBasisV1( xLos, yLos, zLos, theta,
135                                            g, Ng );
136      if ( 0 == pkRealVectorsUnderLineOfSightBasis1( xLos, yLos, zLos, theta,
137                                                     e1, e2, e3,
138                                                     t,
139                                                     x, x1, x2, x3, x12,
140                                                     NULL ) ) {
```

Prepare the Ti*k*Z commands for typesetting the projection of the three-dimensional landscape of $\mathcal{P}(h)$.

```
141         puts(   "\\begin{PkTikzpicture}[scale=1.0," );
142         puts(   "                      mypoint/.style={pktikzpoint,fill=black}," );
143         puts(   "                      %paraboloid/.style={pktikzsurface,shading=axis," );
144         puts(   "                      %                   shading angle=170,opacity=0.5}]" );
145         puts(   "                      paraboloid/.style={draw=pktikzsurfacedrawcolor,top color=w
146         puts(   "                                         bottom color=pktikzsurfacefillcolor," )
147         puts(   "                                         opacity=0.5}]" );
148         puts(   "   %");
149         puts(   "   %\\draw[help lines] (-0.2,-0.2) grid (7.1,5.1);");
150         puts(   "   %");
151         puts(   "   % Some coordinates.");
152         puts(   "   %");
153         puts(   "   \\pktikzSetUncircledPoint{(0,0)}{origin};" );
154         printf( "   \\pktikzSetUncircledPoint{(%g,%g)}{t};\n",
155                 pkRealVectorGetComponent(t)[0], pkRealVectorGetComponent(t)[1] );
156         printf( "   \\pktikzSetUncircledPoint{(%g,%g)}{x};\n",
157                 pkRealVectorGetComponent(x)[0], pkRealVectorGetComponent(x)[1] );
158         puts(   "   %");
159         puts(   "   % Basisvectors.");
160         puts(   "   %");
161         printf( "   \\draw[pktikzbasisvector,<->]\n"
162                 "      (%g,%g) node[below left] {$%s$} --\n"
```

```
163             "      (origin) -- (%g,%g) node[right] {$%s$};\n",
164             pkRealVectorGetComponent(e1)[0],
165             pkRealVectorGetComponent(e1)[1],
166             pkRealVectorGetName(e1),
167             pkRealVectorGetComponent(e2)[0],
168             pkRealVectorGetComponent(e2)[1],
169             pkRealVectorGetName(e2) );
170         printf( "   \\draw[pktikzbasisvector,->]\n"
171             "      (origin) -- (%g,%g) node[above] {$%s$};\n",
172             pkRealVectorGetComponent(e3)[0],
173             pkRealVectorGetComponent(e3)[1],
174             pkRealVectorGetName(e3) );
175
176         puts(   "   %");
177         puts(   "   % The components of the 'x' position.");
178         puts(   "   %");
179         printf( "   \\draw[pktikzdimension,semithick]\n"
180             "      (origin) --\n"
181             "      (%g,%g) --\n"
182             "      (%g,%g) --\n"
183             "      (%g,%g) node[left]{$%s$}\n"
184             "      (%g,%g) node[left=3pt]{$%s$} --\n"
185             "      (%g,%g) --\n"
186             "      (%g,%g) node[above]{$%s$};\n",
187             pkRealVectorGetComponent(x12)[0],
188             pkRealVectorGetComponent(x12)[1],
189             pkRealVectorGetComponent(x)[0],
190             pkRealVectorGetComponent(x)[1],
191             pkRealVectorGetComponent(x3)[0],
192             pkRealVectorGetComponent(x3)[1],
193             pkRealVectorGetName(x3),
194             pkRealVectorGetComponent(x1)[0],
195             pkRealVectorGetComponent(x1)[1],
196             pkRealVectorGetName(x1),
197             pkRealVectorGetComponent(x12)[0],
198             pkRealVectorGetComponent(x12)[1],
199             pkRealVectorGetComponent(x2)[0],
200             pkRealVectorGetComponent(x2)[1],
201             pkRealVectorGetName(x2) );
202
203         puts(   "   %");
204         puts(   "   % The paraboloid.");
205         puts(   "   %");
206         //puts( "   \\draw[pktikztranslucentsurface]" );
207         //puts( "   \\draw[pktikzsurface]" );
208         puts( "   \\draw[paraboloid]" );
209         for ( j = 0;  j < Ny;  j++ ) {
210            printf( "      (%g,%g) --\n",
211                    pkRealVectorGetComponent(r[Nx-1][j])[0],
212                    pkRealVectorGetComponent(r[Nx-1][j])[1] );
213         }
214         for ( i = Nx - 1;  i >= 0;  i-- ) {
215            printf( "      (%g,%g) --\n",
216                    pkRealVectorGetComponent(r[i][Ny-1])[0],
217                    pkRealVectorGetComponent(r[i][Ny-1])[1] );
218         }
219         for ( j = Ny - 1;  j >= 0;  j-- ) {
220            printf( "      (%g,%g) --\n",
221                    pkRealVectorGetComponent(r[0][j])[0],
222                    pkRealVectorGetComponent(r[0][j])[1] );
223         }
224         for ( i = 0;  i < Nx - 1;  i++ ) {
```

```
225            printf( "      (%g,%g) --\n",
226                    pkRealVectorGetComponent(r[i][0])[0],
227                    pkRealVectorGetComponent(r[i][0])[1] );
228         }
229         printf( "      (%g,%g);\n",
230                 pkRealVectorGetComponent(r[Nx-1][0])[0],
231                 pkRealVectorGetComponent(r[Nx-1][0])[1] );
232         printf( "   \\path (%g,%g) node[above right,pktikzsurfacedrawcolor]{$\\parab$};\n",
233                    pkRealVectorGetComponent(r[0][3*Ny/7])[0],
234                    pkRealVectorGetComponent(r[0][3*Ny/7])[1] );
235         puts(   "   %");
236
237         for ( i = 1;  i < Nx - 1;  i++ ) {
238            puts( "   \\draw[pktikzsurfacelines]" );
239            for ( j = 0;  j < Ny;  j++ ) {
240               printf( "      (%g,%g)%s\n",
241                       pkRealVectorGetComponent(r[i][j])[0],
242                       pkRealVectorGetComponent(r[i][j])[1],
243                       ( j < Ny - 1 ) ? " --" : ";" );
244            }
245         }
246         puts( "   %");
247         for ( j = 1;  j < Ny - 1;  j++ ) {
248            puts( "   \\draw[pktikzsurfacelines]" );
249            for ( i = 0;  i < Nx;  i++ ) {
250               printf( "      (%g,%g)%s\n",
251                       pkRealVectorGetComponent(r[i][j])[0],
252                       pkRealVectorGetComponent(r[i][j])[1],
253                       ( i < Nx - 1 ) ? " --" : ";" );
254            }
255         }
256
257         puts(   "   %");
258         puts(   "   % Apex position 't'.");
259         puts(   "   %");
260         printf( "   \\draw[pktikzbasisvector,->]\n"
261                 "      (t) -- (%g,%g);\n",
262                 pkRealVectorGetComponent(e3)[0],
263                 pkRealVectorGetComponent(e3)[1] );
264         printf( "   \\path (t) coordinate[mypoint,label=left:$%s$];\n",
265                 pkRealVectorGetName(t) );
266         puts(   "   %");
267         puts(   "   % 'z'-contour path 'c'.");
268         puts(   "   %");
269         puts(   "   \\draw[pktikzemphvectorcolor] plot[smooth] coordinates {" );
270         for ( i = 0;  i < Nc;  i++ ) {
271            printf( "      (%g,%g)%s\n",
272                    pkRealVectorGetComponent(c[i])[0],
273                    pkRealVectorGetComponent(c[i])[1],
274                    ( i < Nc - 1 ) ? "" : " };" );
275         }
276         printf( "   \\path (%g,%g) node[below,pktikzemphvectorcolor]{$\\vecc(\\sigma;z)$};\n",
277                    pkRealVectorGetComponent(c[Nc-2])[0],
278                    pkRealVectorGetComponent(c[Nc-2])[1] );
279         puts(   "   %");
280         puts(   "   % 'x'-gradient path 'g'.");
281         puts(   "   %");
282         puts(   "   \\draw[pktikzemphvectorcolor] plot[smooth] coordinates {" );
283         for ( i = 0;  i < Ng;  i++ ) {
284            printf( "      (%g,%g)%s\n",
285                    pkRealVectorGetComponent(g[i])[0],
286                    pkRealVectorGetComponent(g[i])[1],
```

```
                    ( i < Ng -1 ) ? "" : " };" );
        }
        printf( "   \\path (%g,%g) node[right=1ex,pktikzemphvectorcolor]{$\\vecg(\\gamma;x,y)$};\
                    pkRealVectorGetComponent(g[4])[0],
                    pkRealVectorGetComponent(g[4])[1] );

        puts(   "   %");
        puts(   "   % Position vector 'x'.");
        puts(   "   %");
        printf( "   \\path (x) coordinate[mypoint,label=below right:$%s$];\n",
                pkRealVectorGetName(x) );
        puts(   "\\end{PkTikzpicture}");

    } else {

        puts("ERROR: 'pkRealVectorsUnderLineOfSightBasis1()' failed.");

    }
```

Finally, clean up.

```
    pkRealVectorFree1(e1);
    pkRealVectorFree1(e2);
    pkRealVectorFree1(e3);
    pkRealVectorFree1(t);
    for ( i = 0;  i < Nx;  i++ ) {
       for ( j = 0;  j < Ny;  j++ ) {
          pkRealVectorFree1(r[i][j]);
       }
    }
    pkRealVectorFree1(x);
    pkRealVectorFree1(x1);
    pkRealVectorFree1(x2);
    pkRealVectorFree1(x3);
    pkRealVectorFree1(x12);
    for ( i = 0;  i < Nc;  i++ )
       pkRealVectorFree1(c[i]);
    for ( i = 0;  i < Ng;  i++ )
       pkRealVectorFree1(g[i]);

    return;
}

/*-------------------------------------------------------------*/

int main( const int argc, const char *argv[] )
{
   _diagram();
   exit(0);
}
```

This simple UNIX "makefile" captures the necessary file dependencies, and demonstrates how to compile the C files.

Generic Make targets.

```
all: embedded-surfaces-in-R3.pdf

clobber: latexclobber
```

```
4           @rm -f *.o
5           @rm -f *.run paraboloid.tex
6           @rm -f *.core
7
8   backup: clobber
9           @PACKDIR=`basename \`pwd\``  &&  cd ..  &&  tar -czvf ${TARPATH} $${PACKDIR}
```

File based Make targets.

```
10  embedded-surfaces-in-R3.pdf: paraboloid.tex Makefile.demo embedded-surfaces-in-R3.bib
```

Implicit rule targets.

```
11  .SUFFIXES: .c .o .run .tex
12  .c.o:
13          clang -c -DDEBUG=2 -I/usr/local/pklib/include -DFreeBSD -o ${@} ${<}
14  .o.run:
15          clang -DDEBUG=2 -I/usr/local/pklib/include -DFreeBSD -o ${@} ${<} \
16                /usr/local/pklib/lib/libpk.a \
17                /usr/local/pklib/lib/libpkmath.a \
18                -lm
19  .run.tex:
20          ./${<} > ${@}
```

Incorporate PKLATEXMAKE.[5]

```
21  # Added by 'pklatexmake.mk'.  Do not delete. 26Jul16
22  .include "/usr/local/pklatexmake/lib/pklatexmake.mk"
```

References

[1] Murray R. Spiegel. *Schaum's Outline Series—Mathematical Handbook of Formulas and Tables*. Number 07-060224-7. McGraw–Hill, 1968.

[2] Paul Kotschy. The PKLIB C software library.

[3] Paul Kotschy. PKTECHDOC: Literate programming for non-TEX programmers. *Still to be published.*

[4] Paul Kotschy. Rotational transformations in three dimensions. *Still to be published.*

[5] Paul Kotschy. The PKLATEXMAKE package. *Still to be published.*