Chapter 2

Details on the numerical solution of the eikonal equation

2.1 Some notes on Eikonal equations and their resolution by fast methods on unstructured grids

First of all, let's consider the problem in its almost general form

$$\begin{cases} H(\mathbf{x}, \nabla u(\mathbf{x})) = 1 & \mathbf{x} \in \Omega \\ u(\mathbf{x}) = g(\mathbf{x}) & \mathbf{x} \in \Gamma \subset \partial\Omega \end{cases} \tag{2.1}$$

where, in our case,

$$H(\mathbf{x}, \nabla u(\mathbf{x})) = |\nabla u(\mathbf{x})|_M = \sqrt{\nabla u(\mathbf{x})^T M(\mathbf{x}) \nabla u(\mathbf{x})}, \tag{2.2}$$

where M is a symmetric positive definite tensor. Note that this formulation includes both anisotropic Eikonal equations, where M is not proportional to the identity, or front speed different from 1 since

$$|\nabla u(\mathbf{x})| = F(\mathbf{x})$$

can be put in the form (2.1) by setting $M = F^{-1}\mathbf{I}$.

2.1.1 Some basic facts

We recall some basic facts and terminology since it helps to understand the problem and also the algorithm for its solution. First of all, we can give u the meaning of the travel time. Wavefronts of the travel time are indeed level sets of u:

$$L_t = \{\mathbf{x} \in \Omega : u(\mathbf{x}) = t, t \in \mathbb{R}\}$$

is the position of the front at "time" t. Consequently the wavefront normal is

$$\mathbf{n} = \frac{\nabla u}{|\nabla u|}, \tag{2.3}$$

which, in the classic case $M = \mathbf{I}$, becomes simply $\mathbf{n} = \nabla u$. Indeed, the vector $\mathbf{p} = \nabla u$, which is in general different from $\mathbf{n}$, is called the *slowness vector* since

$$v_u = \frac{1}{|\mathbf{p}|} = \frac{1}{|\nabla u|}, \tag{2.4}$$

is the *phase speed*, and $v_u(\mathbf{x})$ is the speed at which the front at point $\mathbf{x}$ is moving in the direction $\mathbf{n}$.

2.1.2 The characteristics

The previous definitions allow us to write our problem as

$$H(\mathbf{x}, \mathbf{p}) = \sqrt{\mathbf{p}^T M(\mathbf{x}) \mathbf{p}}.$$

A characteristics is a line $\mathbf{x} = \mathbf{y}(t)$ function of a single parameter t so that

$$\frac{d}{dt} H(\mathbf{y}(t), \mathbf{p}(t)) = \frac{\partial H}{\partial \mathbf{p}} \cdot \frac{d\mathbf{p}}{dt} + \frac{\partial H}{\partial \mathbf{x}} \cdot \frac{d\mathbf{y}}{dt} = 0$$

We have then

$$\begin{cases} \dfrac{d\mathbf{y}}{dt} = \dfrac{\partial H}{\partial \mathbf{p}} = \dfrac{1}{H} M\mathbf{p} = M\mathbf{p} \\ \dfrac{d\mathbf{p}}{dt} = -\dfrac{\partial H}{\partial \mathbf{x}} \\ \dfrac{du}{dt} = \nabla u \cdot \dfrac{d\mathbf{y}}{dt} = \mathbf{p} \cdot \dfrac{d\mathbf{y}}{dt} = \mathbf{p} \cdot \dfrac{\partial H}{\partial \mathbf{p}} = H = 1 \end{cases} \tag{2.5}$$

The last expression comes from the fact that H is homogeneous of order 1 with respect to $\mathbf{p}$. Those expressions tell us that the characteristic line, which is the line along which the information is traveling, has direction $\frac{M\mathbf{p}}{|M\mathbf{p}|}$, and is equal to $\mathbf{n}$ only if $M = \alpha \mathbf{I}$ for a $\alpha > 0$. If M is not proportional to the identity (anisotropic case), the characteristic direction is not $\mathbf{n}$. But, since M is s.p.d., it always forms an angle less than $\pi/2$ with $\mathbf{n}$. The quantity

$$v_g = |\frac{d\mathbf{y}}{dt}| = |M\mathbf{p}| = |M\nabla u|$$

is called *group speed*. If the system is isotropic and homogeneous, $M = F^{-1}\mathbf{I}$ with F constant, we have that $H = H(\mathbf{p})$, and consequently $\dfrac{d\mathbf{p}}{dt} = \mathbf{0}$.

2.1.3 Fast algorithms for unstructured meshes

Fast algorithms for the Eikonal equation (2.1) can be split into two categories, fast sweeping and fast marching. In both methods you somehow follow an advancing front of active nodes, and both methods require the knowledge

of both mesh elements that contains a given node (1-level element neighborhood) and of the one-level node neigborhood (the nodes belonging to the1-level element neighborhood, apart the given node).

At the start all nodes in γ are set as *considered* and all the neighbor nodes set as *active*. The values of the active nodes is updated following a procedure that we well detail later. An important aspect of the updating procedure is that eventually we need to guarantee the *causality principle*, which basically states that the values cannot descrease and the value at a point $\mathbf{x}$ can depend only on points along the characteristics in the "past' direction. A property equivalent to upwinding.

The difference between the two techniques then lays in how the causality principle is satisfied. In the fast marching, we need to keep track of the active node with the smallest value of u, in the fast sweeping this is not required, but several checks are in order to verify the condition. The fast marching has $O(N \log N)$ complexity, because of the need of using a heap, while the fase sweeping are $O(N)$. Moreover, fast sweeping are parallelized more easily (we have also GPU implementation). However, the constants are not the same: for relatively small meshes the fast matching may outperform the fast sweeping counterpart. Of course, most also depends on the efficiency of the data structures used.

2.1.4 The local problem

In both FMM and FSM the key is the resolution of the so called *local problem*. The local problem is able to give a possible update for the value of u at a node. Let's consider two possible variants, in the basic setting of an acute tetrahedron (we wil discuss also the case to triangles and edges).

We give the description for a 3D problem. The 2D case can be derived very easily by reducing the dimensionality. We use the following notation. We indicate with $\mathbf{x}_1, \ldots, \mathbf{x}_4$ the nodes of a generic tetrahedron and with $u_j, j = 1, \ldots, 4$ the current value of u at the corresponding node. We assume that the values of u at nodes 1 to 3 is known and we need to update the value u_4. We also set $\mathbf{e}_{ij} = \mathbf{x}_j - \mathbf{x}_i$ and $u_{ij} = u_j - u_i$. Finally, with f_i we denote the face (edge for triangles) opposite to node $\mathbf{x}_i$.

The local problem can be constructed in different ways, one is that proposed in [SV00], which is basically based on fixing the gradient of u so that it satisfies the Eikonal equation. However, this technique is less flexible than the one proposed in [FKW13, FJP$^+$11], later extended in [GHZ18] and [KSC$^+$07]. So here we refer only to the latter techniques, giving two possible strategy for the resolution.

Constructing the local problem

The method starts by assuming that the foot the characteristics passing through $\mathbf{x}_4$ intersects the face f_4 at an unknown point $\mathbf{x}_5 \in f_4$. Point x_5 may be expressed as function of two unknowns, λ_1 and λ_2

$$\mathbf{x}_5 = \lambda_1 \mathbf{e}_{31} + \lambda_2 \mathbf{e}_{32} + \mathbf{x}_3. \tag{2.6}$$

Analogously, we may set

$$u_5 = \lambda_1 u_{31} + \lambda_2 u_{32} + u_3 = \boldsymbol{\lambda}^T \boldsymbol{\delta u}, \tag{2.7}$$

where

$$\boldsymbol{\lambda} = \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ 1 \end{bmatrix}, \tag{2.8}$$

and

$$\boldsymbol{\delta u} = \begin{bmatrix} u_{31} \\ u_{32} \\ u_3 \end{bmatrix}$$

Note that the condition $\mathbf{x}_5 \in f_4$ is now expressed by $0 \leq \lambda_1 \leq 1$ and $0 \leq \lambda_2 \leq 1$.

We assume that M is constant on the given element (and typically equal to the value taken at $\mathbf{x}_4$), so that we may write, since $du/dt = 1$ along the characteristics,

$$u_4 = u_5 + \sqrt{\mathbf{e}_{54}^T M \mathbf{e}_{54}}$$

Now

$$\mathbf{e}_{54} = \mathbf{x}_4 - \mathbf{x}_5 = -\lambda_1 \mathbf{e}_{31} - \lambda_2 \mathbf{e}_{32} + \mathbf{e}_{34} = \mathbf{E}\boldsymbol{\lambda},$$

where

$$\mathbf{E} = \begin{bmatrix} \mathbf{e}_{13} & \mathbf{e}_{23} & \mathbf{e}_{34}. \end{bmatrix} \tag{2.9}$$

Therefore,

$$u_4 = \boldsymbol{\lambda}^T \boldsymbol{\delta u} + \sqrt{\boldsymbol{\lambda}^T \mathbf{E}^T M \mathbf{E} \boldsymbol{\lambda}} = \boldsymbol{\lambda}^T \boldsymbol{\delta u} + \sqrt{\boldsymbol{\lambda}^T \widehat{\mathbf{M}} \boldsymbol{\lambda}} = \Psi(\lambda_1, \lambda_2), \tag{2.10}$$

with

$$\widehat{\mathbf{M}} = \mathbf{E}^T M \mathbf{E}.$$

Note that $\widehat{\mathbf{M}}$ is s.p.d. Indeed, M is s.p.d. by hypothesis, so it is sufficient to show that $\mathbf{E}$ has full rank. And this is the case since $\det(\mathbf{E}) \neq 0$, being proportional to the volume of the tetrahedron (or the area of triangle in the 2D case).

Therefore, $\sqrt{\boldsymbol{\lambda}^T\widehat{\mathbf{M}}\boldsymbol{\lambda}} = \|\boldsymbol{\lambda}\|_{\widehat{\mathbf{M}}}$ is a norm. Consequently Ψ *is a convex function*, being the sum of a linear function and a convex function. Indeed, thanks to the triangle inequality, for any $\alpha \in [0,1]$ we have,

$$\Psi(\alpha\lambda_1+(1-\alpha)\beta_1, \alpha\lambda_2+(1-\alpha)\beta_2) = (\alpha\boldsymbol{\lambda}^T+(1-\alpha)\boldsymbol{\beta}^T)\boldsymbol{\delta u}+\|\alpha\boldsymbol{\lambda}+(1-\alpha)\boldsymbol{\beta}\|_{\widehat{\mathbf{M}}} \leq$$
$$\alpha\boldsymbol{\lambda}^T\boldsymbol{\delta u}+(1-\alpha)\boldsymbol{\beta}^T\boldsymbol{\delta u}+\alpha\|\boldsymbol{\lambda}\|_{\widehat{\mathbf{M}}}+((1-\alpha)\|\boldsymbol{\beta}\|_{\widehat{\mathbf{M}}} = \alpha\Psi(\lambda_1,\lambda_2)+(1-\alpha)\Psi(\beta_1,\beta_2).$$

Not only, we have that for $\alpha \in (0,1)$ and $\lambda_i \neq \beta_i$

$$\Psi(\alpha\lambda_1 + (1-\alpha)\beta_1, \alpha\lambda_2 + (1-\alpha)\beta_2) < \alpha\Psi(\lambda_1,\lambda_2) + (1-\alpha)\Psi(\beta_1,\beta_2),$$

since under those conditions $\|\alpha\boldsymbol{\lambda} + (1-\alpha)\boldsymbol{\beta}\|_{\widehat{\mathbf{M}}} = \alpha\|\boldsymbol{\lambda}\|_{\widehat{\mathbf{M}}} + (1-\alpha)\|\boldsymbol{\beta}\|_{\widehat{\mathbf{M}}}$ only if $\boldsymbol{\lambda} = c\boldsymbol{\beta}$ for a $c \neq 0$. But this is impossible since the third component of those two vectors is 1 by construction. It means that Ψ is *strictly convex*. Moreover, we will show later that for the allowable value of λ_i the Hessian exists and is s.p.d. Consequently, Ψ is also *strongly convex*.

The two unknown λs are found by noting that the characteristics is the line of minimal traveling time. So we have to solve the following *constrained minimization problem*

$$u_4 = \operatorname{argmin}_{\substack{0\leq\lambda_1,\lambda_2 \\ \lambda_1+\lambda_2\leq 1}} \Psi(\lambda_1,\lambda_2). \tag{2.11}$$

We have the following result: u_4 solution of (2.11) satisfies $u_4 > \min\{u_j,\ j = 1,\dots,3\}$. Indeed, we can always order the nodes of the tetrahedron so that $u_3 \leq u_2 \leq u_1$. This way, we must prove that $u_4 - u_3 > 0$. Indeed,

$$u_4 - u_3 = \lambda_1 u_{31} + \lambda_2 u_{32} + \sqrt{\boldsymbol{\lambda}^T\widehat{\mathbf{M}}\boldsymbol{\lambda}} > 0,$$

since it is the sum of two non negative numbers (due to the hypothesis and the constraints on the λs) and a positive number (due to the fact that $\boldsymbol{\lambda} \neq \mathbf{0}$ and $\widehat{\mathbf{M}}$ is s.p.d.).

Some notes

In the following we will denote with $\overline{\boldsymbol{\lambda}} = \begin{bmatrix}\lambda_1\\ \lambda_2\end{bmatrix}$ the vector of the actual unkwnowns, and with $\overline{\boldsymbol{\delta u}}$ the matrix $\begin{bmatrix}u_1-u_3\\ u_2-u_3\end{bmatrix}$. Moreover, we can write matrix $\widehat{\mathbf{M}}$ in block form

$$\widehat{\mathbf{M}} = \begin{bmatrix}\overline{\mathbf{M}} & \mathbf{a}\\ \mathbf{a}^T & b\end{bmatrix}, \tag{2.12}$$

where in 3D $\overline{\mathbf{M}}$ and $\mathbf{a}$ are a 2×2 s.p.d. tensor and a two dimensional vector, respectively, while in 2D they are just two real numbers, with $\overline{M} > 0$. With this notation we have

$$\Psi = u_3 + \overline{\boldsymbol{\delta u}}^T\overline{\boldsymbol{\lambda}} + \sqrt{b + 2\mathbf{a}^T\overline{\boldsymbol{\lambda}} + \overline{\boldsymbol{\lambda}}^T\overline{\mathbf{M}\boldsymbol{\lambda}}}, \tag{2.13}$$

which in 2D reduces to

$$\Psi = u_2 + (u_1 - u_2)\lambda_1 + \sqrt{b + 2a\lambda_1 + \overline{M}\lambda_1^2}. \tag{2.14}$$

We recall also the Cauchy-Schwartz inequality,

$$\boldsymbol{\lambda}^T \widehat{\mathbf{M}} \boldsymbol{\beta} = (\boldsymbol{\lambda}, \boldsymbol{\beta})_{\widehat{\mathbf{M}}} \leq \|\boldsymbol{\lambda}\|_{\widehat{\mathbf{M}}} \|\boldsymbol{\beta}\|_{\widehat{\mathbf{M}}}, \ \forall \boldsymbol{\lambda}, \boldsymbol{\beta} \in \mathbb{R}^d. \tag{2.15}$$

2.1.5 Possible procedures

The first possibility, advocated in the cited literature, is to explicitly solve $d\Psi/d\boldsymbol{\lambda} = \mathbf{0}$, which gives rise to a quadratic system, and check a-posteriori the constraints. Let's first recall this possibility.

Solving a quadratic system

$$d\Psi/d\boldsymbol{\lambda} = \begin{bmatrix} u_{31} + \dfrac{\mathbf{e}_{13} \cdot M(\lambda_1 \mathbf{e}_{13} + \lambda_2 \mathbf{e}_{23} + \mathbf{e}_{34})}{\sqrt{\boldsymbol{\lambda}^T \widehat{\mathbf{M}} \boldsymbol{\lambda}}} \\ u_{32} + \dfrac{\mathbf{e}_{23} \cdot M(\lambda_1 \mathbf{e}_{13} + \lambda_2 \mathbf{e}_{23} + \mathbf{e}_{34})}{\sqrt{\boldsymbol{\lambda}^T \widehat{\mathbf{M}} \boldsymbol{\lambda}}} \end{bmatrix}. \tag{2.16}$$

Setting to zero brings

$$\begin{cases} u_{31}\sqrt{\boldsymbol{\lambda}^T \widehat{\mathbf{M}} \boldsymbol{\lambda}} + \mathbf{e}_{13} \cdot M(\lambda_1 \mathbf{e}_{13} + \lambda_2 \mathbf{e}_{23} + \mathbf{e}_{34}) = 0 \\ u_{32}\sqrt{\boldsymbol{\lambda}^T \widehat{\mathbf{M}} \boldsymbol{\lambda}} + \mathbf{e}_{23} \cdot M(\lambda_1 \mathbf{e}_{13} + \lambda_2 \mathbf{e}_{23} + \mathbf{e}_{34}) = 0 \end{cases} \tag{2.17}$$

In the case $u_{31} \neq 0$ and $u_{32} \neq 0$ one of the two equations may be replaced by

$$u_{32}\mathbf{e}_{13}^T M(\lambda_1 \mathbf{e}_{13} + \lambda_2 \mathbf{e}_{23} + \mathbf{e}_{34}) - u_{31}\mathbf{e}_{23}^T M(\lambda_1 \mathbf{e}_{13} + \lambda_2 \mathbf{e}_{23} + \mathbf{e}_{34}) = 0. \tag{2.18}$$

Note that, in the case $M = F^{-2}\mathbf{I}$, where F is the front speed, (2.17) becomes simply

$$\begin{cases} u_{31}F\sqrt{\boldsymbol{\lambda}^T \mathbf{E}^T \mathbf{E} \boldsymbol{\lambda}} + (\lambda_1 |\mathbf{e}_{13}|^2 + \lambda_2 \mathbf{e}_{13} \cdot \mathbf{e}_{23} + \mathbf{e}_{13} \cdot \mathbf{e}_{34}) = 0 \\ u_{32}F\sqrt{\boldsymbol{\lambda}^T \mathbf{E}^T \mathbf{E} \boldsymbol{\lambda}} + (\lambda_1 \mathbf{e}_{23} \cdot \mathbf{e}_{13} + \lambda_2 |\mathbf{e}_{23}|^2 + \mathbf{e}_{23} \cdot \mathbf{e}_{34}) = 0 \end{cases} \tag{2.19}$$

and (2.18) is now

$$u_{32}(\lambda_1 |\mathbf{e}_{13}|^2 + \lambda_2 \mathbf{e}_{13} \cdot \mathbf{e}_{23} + \mathbf{e}_{13} \cdot \mathbf{e}_{34}) - u_{31}(\lambda_1 \mathbf{e}_{23} \cdot \mathbf{e}_{13} + \lambda_2 |\mathbf{e}_{23}|^2 + \mathbf{e}_{23} \cdot \mathbf{e}_{34}) = 0.$$

which may replace one of (2.19).

Those non-linear problems are of quadratic nature (it is sufficient to square the terms) and may have multiple solution, to be treated carefully. Details in [FJP$^+$11, FKW13]. Operative tools on how to speed-up computations, also in parallel architectures, can be found in [GHZ18].

Minimization problem

An alternative technique for the solution of problem (2.11) is indeed to use an optimization algorithm for constrained optimization, for instance projected Newton or BFGS. This is the route taken in [CdV22], where they have referred to the fast marching algorithm in [KSC$^+$07] specially designed for highly anisotropic problems.

It may seem a cumbersome procedure compare to set to zero the derivatives. Indeed, the computational cost is probably higher. However, we have some advantages:

- Ψ is strictly convex and the constraint set is a bounded convex set (it's a triangle in 3D, a line in 2D). Therefore, *the solution of our constrained minimization problem exists and is unique*;
- Solving (2.17) is often done by squaring the elements to get a quadratic expression, and this may introduce spurious roots. Moreover, in the 3D case you still have a quadratic system to solve!
- Since the minimization is constrained, it seems that the system may solve the particular solutions linked to $\mathbf{x}_5$ falling outside f_4 automatically. Assume that the minimization is leading λ_1 towards negative values. A box constrained algorithm will then force $\lambda_1 = 0$ in the minimization procedure. But this is equivalent to search for the foot $\mathbf{x}_5$ on edge $2-3$ of the tetrahedron. It is then like minimizing on the face f_1 formed by nodes 2, 3 and 4, which is what we would do with the standard algorithm to treat this case! Let assume that also λ_2 is driven by the descent algorithm to be negative. Then the constraint will force also $\lambda_2 = 0$. Therefore, the minimum is given by

  $$\mathbf{x}_5 = \mathbf{x}_3,$$

  and

  $$u_4 = u_3 + \sqrt{\mathbf{e}_{34} M \mathbf{e}_{34}},$$

  exactly what we expect also with the standard algorithm. Therefore, the minimization algorithm may be set to treat these special cases directly.

The optimization algorithm. To activate the optimization we need first to compute the derivatives, since we want to use a second order method for efficiency. To the purpose we introduce the following quantities. We have the following expression for the gradient with respect to (λ_1, λ_2), which is just a reformulation of (2.16)

$$\nabla_\lambda \Psi(\lambda_1, \lambda_2) = \overline{\delta \boldsymbol{u}} + \frac{\mathcal{I}\widehat{\mathbf{M}}\boldsymbol{\lambda}}{\|\boldsymbol{\lambda}\|_{\widehat{\mathbf{M}}}} \tag{2.20}$$

where we recall that $\boldsymbol{\lambda}$ is defined in (2.8). This formula allows to express the gradient via matrix-vector operations, we also note that pre-multiplying a matrix by $\mathcal{I}$ is equivalent to taking the first two rows.

We can also compute the Hessian:

$$\nabla^2_\lambda \Psi(\lambda_1, \lambda_2) = \frac{\mathcal{I}\widehat{\mathbf{M}}\mathcal{I}^T}{\|\boldsymbol{\lambda}\|_{\widehat{\mathbf{M}}}} - \frac{\mathcal{I}\widehat{\mathbf{M}}\boldsymbol{\lambda}\boldsymbol{\lambda}^T\widehat{\mathbf{M}}\mathcal{I}^T}{\|\boldsymbol{\lambda}\|^3_{\widehat{\mathbf{M}}}} \tag{2.21}$$

Note: Remember that in fact, in the 3D case the gradient is a vector of 2 components and the Hessian a 2×2 matrix. While, in the 2D case they are both scalars (the Hessian is in this case a positive number). We keep using the vector notation of the 3D case since it is more general and easily "translated" into the 2D counterpart.

We may also note that $\mathcal{I}\widehat{\mathbf{M}}\mathcal{I}^T$ is nothing else than the upper-left 2×2 block of matrix $\widehat{\mathbf{M}}$. It may be shown that the Hessian is s.p.d. Indeed, let $\mathbf{x} \in \mathbb{R}^2$ be a vector different from zero. We may show that $\mathbf{x}^T \nabla^2_\lambda \Psi \mathbf{x} > 0$ for all λ_1 and λ_2. Indeed, if we call $\hat{\mathbf{x}} = \mathcal{I}^T\mathbf{x}$, we have to show that

$$\|\boldsymbol{\lambda}\|^2_{\hat{\mathbf{M}}}\|\hat{\mathbf{x}}\|^2_{\hat{\mathbf{M}}} - \|\hat{\mathbf{x}}^T\hat{\mathbf{M}}\boldsymbol{\lambda}\|^2 > 0.$$

But this is true since, thanks to Cauchy-Schwartz inequality,

$$\|\hat{\mathbf{x}}^T\hat{\mathbf{M}}\boldsymbol{\lambda}\|^2 = \|\hat{\mathbf{x}}^T\hat{\mathbf{M}}^{1/2}\hat{\mathbf{M}}^{1/2}\boldsymbol{\lambda}\|^2 \leq \|\hat{\mathbf{x}}\|^2_{\hat{\mathbf{M}}}\|\boldsymbol{\lambda}\|^2_{\hat{\mathbf{M}}},,$$

and the equality can be reached only when $\hat{\mathbf{x}}$ and $\boldsymbol{\lambda}$ are proportional (aligned). But this is impossible, since, by construction, the third component of $\boldsymbol{\lambda}$ is 1 and that of $\hat{\mathbf{x}}$ is 0. Therefore, the inequality is in fact strict. And this ends the proof.

The 2D case. In the 2D case the expressions simplifies since now Ψ is a function of a single parameter λ_1. Writing the expression for the Ψ and its derivatives explicitly as function of the parameter we have:

$$\Psi(\lambda_1) = u_2 + (u_1 - u_2)\lambda_1 + \sqrt{\lambda_1^2\hat{m}_{11} + 2\lambda_1\hat{m}_{12} + \hat{m}_{22}} \tag{2.22}$$

$$\frac{d\Psi}{d\lambda_1}(\lambda_1) = (u_2 - u_1) + \frac{\hat{m}_{11}\lambda_1 + \hat{m}_{12}}{\sqrt{\lambda_1^2\hat{m}_{11} + 2\lambda_1\hat{m}_{12} + \hat{m}_{22}}} \tag{2.23}$$

$$\frac{d^2\Psi}{d\lambda_1^2}(\lambda_1) = \frac{\hat{m}_{11}\hat{m}_{22} - \hat{m}_{12}^2}{(\lambda_1^2\hat{m}_{11} + 2\lambda_1\hat{m}_{12} + \hat{m}_{22})^{3/2}} \tag{2.24}$$

We may note that, being $\hat{\mathbf{M}}$ s.p.d., $\hat{m}_{11}\hat{m}_{22} - \hat{m}_{12}^2 = \det(\hat{\mathbf{M}}) > 0$. Therefore, the second derivative is positive.

Projected Newton

To solve problem we need to define the projection operator

$$\Pi(\overline{\boldsymbol{\lambda}}),$$

on the constraint set

$$\{\overline{\boldsymbol{\lambda}} = (\lambda_1, \lambda_2) : \lambda_1 \geq 0, \lambda_2 \geq 0, \lambda_1 + \lambda_2 \leq 1\}.$$

To this purpose, we define the vectors

$$\mathbf{a}_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \ \mathbf{a}_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \ \mathbf{a}_3 = \frac{\sqrt{2}}{2} \begin{bmatrix} -1 \\ -1 \end{bmatrix}, \ \mathbf{a}_3^* = \frac{\sqrt{2}}{2} \begin{bmatrix} -1 \\ 1 \end{bmatrix},$$

and the rotation matrix

$$\mathbf{R} = \left[\mathbf{a}_3^*, \mathbf{a}_3\right] = \frac{\sqrt{2}}{2} \begin{bmatrix} -1 & -1 \\ 1 & -1 \end{bmatrix}.$$

Note that $\mathbf{a}_3 \cdot \mathbf{a}_3^* = 0$.

We also define the following projectors,

$$\pi^{\pm}(x_1, x_2) = (\max(0, \min(x_1, 1)), \max(0, \min(x_2, 1))),$$

and

$$\pi^3(\lambda_1, \lambda_2) = \mathbf{R}^T \mathbf{p}(\mathbf{R}\overline{\boldsymbol{\lambda}})$$

where

$$\mathbf{p}(x_1, x_2) = \begin{bmatrix} \max(-\sqrt{2}/2, x_1) \\ x_2 \end{bmatrix}. \tag{2.25}$$

We may note that the output $\mathbf{y} = (y_1, y_2)$ of $\pi^3(x_1, x_2)$ satisfies $y_1 + y_2 \leq 1$. Indeed, we have, after a few algebraic manipulations,

$$(y_1, y_2) = \pi^3(x_1, x_2) \rightarrow \begin{cases} y_1 & = \frac{1}{2}[\min(1, (x_1 + x_2)) + x_1 - x_2] \\ y_2 & = \frac{1}{2}[\min(1, (x_1 + x_2)) - x_1 + x_2] \end{cases},$$

by which,

$$y_1 + y_2 = \min(1, x_1 + x_2).$$

Then,

$$\Pi = \pi^{\pm} \circ \pi^3,$$

if the projector on the constraints set $\mathcal{C}$.

To solve the minimization problem, we use the projected Newton techniques, as originally proposed in [Bet82] . We indicate with $\overline{\boldsymbol{\lambda}}^{(k)} = [\lambda_1^{(k)}, \lambda_2^{(k)}]^T$ the $k - th$ iterate of the unknowns (note that it comprises only the actual unkknowns), and with Φ^k, $\nabla\Phi^{(k)}$ and $\mathbf{B}^{(k)} = \mathbf{H}^{-1}(\overline{\boldsymbol{\lambda}}^{(k)})$ the value of the

objective function, its gradient and the inverse of the Hessian at the same iterate. Indeed, the projected Newton algorithm requires to have the inverse of the Hessian at disposal, but in our case where the number of dimensions of the problem is at most two the inverse con be found directly with little effort! The 2D case, where the Hessian is just a scalar, is a little special, but the algorithm is still valid also in that case (with the correct interpretation). At every step of the iterative procedure we can identify the following sets (I recall that $d = 2$ or 1 if the problem is 3D or 2D, respectively):

$$\begin{cases} \mathcal{C}^{(k)} = \{i \in \{1, d\} : (\lambda_i^{(k)} = 0 \text{ and } [\nabla\Psi]_i > 0) \text{ or } (\lambda_1^{(k)} + \lambda_2^{(k)} = 1 \text{ and } \nabla\Psi \cdot \mathbf{a}_3 > 0)\}, \\ \mathcal{F}^{(k)} = \{1, d\} \setminus \mathcal{C}^{(k)}. \end{cases} \tag{2.26}$$

The first set is called the *constrained point set*, the second the *free point set*. The constrained point set is indeed formed by the index of the unknowns which are at the boundary of the constraints set with the gradient forming an angle less than $\pi/2$ with the inner normal of the set. The free points set is just the complement. We define the following projection matrices

$$\mathbf{P}_i = \mathbf{I} - \mathbf{a}_i \otimes \mathbf{a}_i, \ i = 1, \dots d + 1.$$

Note that

$$\mathbf{P}_1 = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}, \ \mathbf{P}_2 = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, \ \mathbf{P}_3 = \begin{bmatrix} 1/2 & -1/2 \\ -1/2 & 1/2 \end{bmatrix},$$

and that $\mathbf{P}_1\mathbf{P}_2 = \mathbf{P}_2\mathbf{P}_1 = \mathbf{0}$ is the zero matrix. Note that $\mathbf{P}_i$ projects a vector of $\mathbb{R}^2$ on the space orthogonal to $\mathbf{a}_i$.

Now we consider the following conditions,

$$\begin{aligned} c_i &= (\lambda_i^{(k)} = 0 \text{ and } [\nabla\Psi]_i > 0) \qquad\qquad i = 1, d, \\ c_3 &= (\lambda_1^{(k)} + \lambda_2^{(k)} = 1 \text{ and } \nabla\Psi \cdot \mathbf{a}_3 > 0), \end{aligned}$$

and matrix

$$\mathbf{D}^{(k)} = \begin{cases} \mathbf{0} & \text{if } \begin{cases} c_1 \text{ and } c_2 \\ \lambda_1^{(k)}, \lambda_2^{(k)} = 1, 0 \text{ and } \nabla\Psi \cdot \mathbf{a}_3^* \geq 0 \text{ and } \nabla\Psi \cdot \mathbf{a}_1 \leq 0 \\ \lambda_1^{(k)}, \lambda_2^{(k)} = 0, 1 \text{ and } \nabla\Psi \cdot \mathbf{a}_3^* \leq 0 \text{ and } \nabla\Psi \cdot \mathbf{a}_2 \leq 0 \end{cases} \\ \mathbf{P}_i^T \mathbf{B}^{(k)} \mathbf{P}_i & \text{if } c_i, \ i = 1, 2, 3. \\ \mathbf{B}^{(k)} & \text{otherwise.} \end{cases} \tag{2.27}$$

Note that if the constrained set is empty $\mathbf{D}^{(k)}$ is just the inverse of the Hessian, and, as we will see, in that case the projected Newton step corresponds to a Newton step.

The Algorithm Given $\overline{\boldsymbol{\lambda}}^{(0)} = [1/3, 1/3]^T$ (initial iterate), ϵ (tolerance) and $\sigma \in (0, 0.5)$ (sufficient descent coefficient, typically 0.1), for $k = 0, 1, \ldots$ we do

1. Compute the free and constrained set.
2. Compute $\mathbf{D}^{(k)}$ as in (2.27), and set $\mathbf{z} = -\mathbf{D}^{(k)} \nabla \Psi^{(k)}$ (projected Newton step).
3. Stopping criteria 1. Stop if $\|\mathbf{z}\| \leq \epsilon$, and $\overline{\boldsymbol{\lambda}}^{(k)}$ is the desired approximation of the minimal point.
4. (Simple backtracking) Compute $\alpha_0 = \frac{\|\Pi(\overline{\boldsymbol{\lambda}}^{(k)} + \mathbf{z})\|}{\|\overline{\boldsymbol{\lambda}}^{(k)} + \mathbf{z}\|}$, and the smallest integer $n \geq 0$ for which the step $\alpha_j = 2^{-n}\alpha_0$ satisfies the sufficient decrease condition:
$$\Psi\left(\Pi(\overline{\boldsymbol{\lambda}}^{(k)} + \alpha_j \mathbf{z})\right) < \Psi^{(k)} + \alpha_j \sigma \mathbf{z}^T \nabla \Psi^{(k)}.$$
5. Set $\overline{\boldsymbol{\lambda}}^{(k+1)} = \Pi(\overline{\boldsymbol{\lambda}}^{(k)} + \alpha_j \mathbf{z})$.
6. Stopping criteria 2. Stop if $\|\overline{\boldsymbol{\lambda}}^{(k+1)} - \overline{\boldsymbol{\lambda}}^{(k)}\| \leq \epsilon$, and $\overline{\boldsymbol{\lambda}}^{(k+1)}$ is the desired approximation of the minimal point.

Note that this formulation of the algorithm is a little different from the one in [Bet82], but can be shown to be almost equivalent. The difference is that in the cited paper a box constraint was considered, while here we have a triangular constraints set. Moreover, the inverse Hessian in [Bet82] is modified so that it maintains positive definiteness, while in our case we have a non-negative modified Hessian (some eigenvalues are 0). But this is not a problem because the components set to zero are exactly those that will be annihilated by the projection step.

Moreover, in [Bet82] the definition of the constrained point set is slightly different: the constraint condition is "smoothed" by the presence of a small tolerance. We have found that it is not necessary.

2.2 The full algorithm

To describe the full algorithms, we need to introduce some notation. First, we assume to have a mesh $\mathcal{M}$ of simplexes (triangles in 2D, tetras in 3D) where vertexes are in $\mathcal{V}$, a set of points, $\mathcal{G}$ where we have assigned the initial value for the field variable u, and the function $\mathcal{Z}(p)$ identifying twin points. It either returns an empty set or a point q so that p and q should satisfy the property $u(q) = u(p) + \Delta_{pq}$. We also use the convention that if $q = \mathcal{Z}(p)$ then also $p = \mathcal{Z}(q)$ with $\Delta_{qp} = -\Delta_{pq}$.We use also the following notation (we use the term point and vertex interchangeably)