MSc in Mathematics and Foundations of Computer Science

Dissertation

(Co)limit computations for diagrams of vector spaces

UNIVERSITY OF OXFORD

student 1358975

Kellogg College

Trinity Term 2021

Contents

Abstract

(Co)limits of finite diagrams of finite-dimensional vector spaces appear in several applied and computational contexts. Earlier this year, an efficient method to compute limits of such diagrams was published in [SHN21]. One of my main contributions is to dualise the described method so as to compute colimits. I have also implemented in Python the algorithm and have analysed the code performances.

Introduction

As detailed in section 1, finite diagrams of finite-dimensional vector spaces can be seen as quiver representations. From this point of view, a limit of such a diagram becomes the space of sections.

Many abstract problems can be expressed as a quiver representation, including a whole range of problems from linear algebra. In most cases, the main properties are linked to the decomposability of the associated quiver [BGP73]. However, more recently, quiver representations started to appear in more applied contexts like the study of *cellular sheaves* [Cur14]. More precisely, the 0^{th} sheaf cohomology group is given by the limit of the associated quiver representation.

The goal of this dissertation is to describe, implement and analyse an algorithm to compute limits and colimits (the dual notion) of a quiver representation. We hope that our algorithm will be applied one day in a computational setting. We thus aim to provide an efficient algorithm.

Section 2 describes two dual algorithms to compute limits and colimits of quiver representations. The approach for limits follows [SHN21]; and I obtained the algorithm for colimits by dualising step by step the one for limits. In addition to the description from [SHN21], I considered, in subsection 2.5, the advantages of computing separately each connected component. And, in subsection 2.6, I extended some of the main propositions to other categories.

I then implemented in Python the algorithm described in section 2 (the code is in annex B). Section 3 details the choices of implementation, and compares the performances to those of a more direct and simple approach.

Finally, to understand, adapt and implement the algorithm from [SHN21], elements from various theories like category theory, numerical linear algebra or networks are necessary. The most important ones are summarised in annex A.

1 Quivers, (co)sections and (co)limits

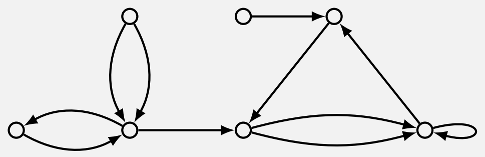
1.1 Quivers

We start by defining the notion of diagrams of vector spaces. From the category theory point of view, they are diagrams of finite shape in the category of finite dimensional vector spaces. In this dissertation we will use a more practical definition by decomposing it into two components: the *quiver* and the *representation*.

Definition 1.1 A *quiver* Q is a finite multi-digraph.
More precisely a quiver Q consists of two finite sets V (the vertices) and E (the edges) together with two functions $s, t : E \to V$ called the source and the target.

Example 1.1

We will use this quiver as our running example. Note that it contains self-loops and multiple edges.



We fix a field $\mathbb{F}$. We denote $\mathrm{Vect}_{\mathrm{Fin-Dim}}$ the category of finite-dimensional vector spaces over $\mathbb{F}$.

Definition 1.2 A *representation* $\mathbf{A}_\bullet$ of a quiver Q is the data of finite-dimensional $\mathbb{F}$-vector spaces $\mathbf{A}_v$ for all $v \in V$ and linear maps $\mathbf{A}_e : \mathbf{A}_{s(e)} \to \mathbf{A}_{t(e)}$ for all $e \in E$.

Remark 1.1 A quiver Q can be seen as a category (see annex A.2) with

- objects $\mathrm{ob}(Q) := V$
- and morphisms $\hom_Q(u,v) := \{e \in E \mid (s(e),t(e)) = (u,v)\}$, (plus the identity when $u = v$).

A representation of Q can be seen as a functor from Q to $\mathrm{Vect}_{\mathrm{Fin-Dim}}$ or alternatively as a Q-shaped diagram in $\mathrm{Vect}_{\mathrm{Fin-Dim}}$.

Definition 1.3 Let $Q = (V,E)$ be a quiver and $\mathbf{A}_\bullet$ a representation of Q. A path p in Q is a list of edges $e_1, \dots, e_m$ with distinct sources such that $s(e_{i+1}) = t(e_i)$ for $1 \leqslant i < m$. We define $t(p) := t(e_m)$, $s(p) := s(e_1)$ and $\mathbf{A}_p := \mathbf{A}_{e_m} \circ \dots \mathbf{A}_{e_1}$. For $v_1, v_2 \in V$, we denote $\mathcal{P}(Q, v_1, v_2)$ or $\mathcal{P}(v_1, v_2)$ the set of all paths in Q from v_1 to v_2.

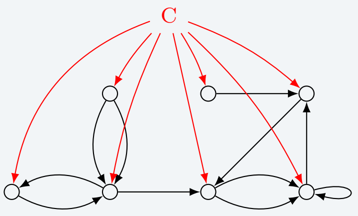
1.2 Sections and limits

Let $Q = (V,E,s,t)$ be a quiver and $\mathbf{A}_\bullet$ be a representation of Q. Once again the notion of limit comes from category theory. We will reformulate it in our particular setting as a certain subspace of the total space $\prod_{v \in V} \mathbf{A}_v$.

Definition 1.4 (Limit).

- A cone of $(Q, \mathbf{A}_\bullet)$ is a finite dimensional vector space C together with linear maps $\phi_v : C \to \mathbf{A}_v$ for $v \in V$, so that, for all $e \in E$, we have the following compatibility requirement:

$$\phi_{t(e)} = \mathbf{A}_e \circ \phi_{s(e)}$$



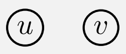
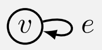
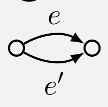
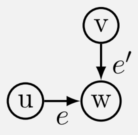
- A limit of $(Q, \mathbf{A}_\bullet)$ is a cone $(C, (\phi_v)_V)$ so that, for all other cones $(C', (\phi'_v)_V)$, there is a unique linear map $\alpha : C' \to C$ such that, for all $v \in V$,

$$\phi'_v = \phi_v \circ \alpha$$

$$\begin{tikzcd}
& C' \arrow[d, "\exists!"] \arrow[ddl, bend right, "\phi'_u"'] \arrow[ddr, bend left, "\phi'_v"] & \\
& C \arrow[dl, "\phi_u"'] \arrow[dr, "\phi_v"] & \\
\mathbf{A}_u \arrow[rr, "\mathbf{A}_e"'] & & \mathbf{A}_v
\end{tikzcd}$$

It follows from this definition that limits are unique up to ismorphisms. Nonetheless, we still allow ourselves to speak of "the" limit, when the property holds for any of the isomorphic limits. Many algebraic constructions can be expressed as limits:

Example 1.2

Construction	Formula	As limit of a diagram
Product	$\mathbf{A}_u \times \mathbf{A}_v$	
Fixed points	$\ker(\mathbf{A}_e - \mathrm{id}_{\mathbf{A}_v})$	
Equalizer	$\mathrm{Eq}(\mathbf{A}_e, \mathbf{A}_{e'}) := \ker(\mathbf{A}_e - \mathbf{A}_{e'})$	
Pullback	$\mathbf{A}_u \times_{\mathbf{A}_w} \mathbf{A}_v :=$ $\{(a,b) \in \mathbf{A}_u \times \mathbf{A}_v \mid \mathbf{A}_e(a) = \mathbf{A}_{e'}(b)\}$	

The problem of the definitions above is that cones and limits are defined up to isomorphism: our algorithm will need to choose one of the isomorphic limits to output. Using the map $\theta \begin{cases} C & \to \prod_{v\in V} \mathbf{A}_v \\ x & \mapsto (\phi_v(x))_{v\in V} \end{cases}$ for every cone $(C, (\phi_v)_V)$, we will show that we can choose the limit to be a subspace of $\prod_{v\in V} \mathbf{A}_v$ together with the projection maps $(\pi_v : \prod_{u\in V} \mathbf{A}_u \to \mathbf{A}_v)_{v\in V}$.

$$\begin{tikzcd}
 & \theta(C) \arrow[dr, "\pi_v"] & \\
C \arrow[ur, "\theta"] \arrow[rr, "\phi_v"'] & & \mathbf{A}_v
\end{tikzcd}$$

Definition 1.5 A section γ associated with Q and $\mathbf{A}_\bullet$ is an element $(\gamma_v)_{v\in V} \in \prod\limits_{v\in V} \mathbf{A}_v$ such that for every edge $e \in E$, we have the following compatibility requirement: $\mathbf{A}_e(\gamma_{s(e)}) = \gamma_{t(e)}$. The set of all sections is denoted $\Gamma(Q, \mathbf{A}_\bullet)$.

Proposition 1.1

$$\Gamma(Q, \mathbf{A}_\bullet) = \bigcap_{e\in E} \ker\left(\pi_{t(e)} - \mathbf{A}_e \circ \pi_{s(e)}\right).$$

Proof. Let $\gamma := (\gamma_v)_{v\in V} \in \prod_{v\in V} \mathbf{A}_v$.

$$\begin{aligned} \gamma \in \Gamma(Q, \mathbf{A}_\bullet) &\iff \forall e \in E,\ \mathbf{A}_e(\pi_{s(e)}(\gamma)) - \pi_{t(e)}(\gamma) = 0 \\ &\iff \forall e \in E,\ \gamma \in \ker\left(\pi_{t(e)} - \mathbf{A}_e \circ \pi_{s(e)}\right) \end{aligned}$$

□

Corollary 1.1 $\Gamma(Q, \mathbf{A}_\bullet)$ is a subspace of $\prod_{v\in V} \mathbf{A}_v$.

Proposition 1.2 $\Gamma(Q, \mathbf{A}_\bullet)$ together with the restrictions of the projection maps $(\pi_v)_{v\in v}$ is a limit of the Q-shaped diagram corresponding to $\mathbf{A}_\bullet$.

Proof.

For every $e \in E$, the compatibility condition of a section translates into the following commutative diagram:

$$\begin{tikzcd}
 & \Gamma(Q, \mathbf{A}_\bullet) \arrow[dl, "\pi_{s(e)}"'] \arrow[dr, "\pi_{t(e)}"] & \\
\mathbf{A}_{s(e)} \arrow[rr, "\mathbf{A}_e"] & & \mathbf{A}_{t(e)}
\end{tikzcd}$$

Hence, $\Gamma(Q, \mathbf{A}_\bullet)$ is a cone. Moreover, if $(C, (\phi_v)_{v \in V})$ is another cone, $\theta(C) \subseteq \Gamma(Q, \mathbf{A}_\bullet)$ and for a linear map $\alpha : C \to \Gamma(Q, \mathbf{A}_\bullet)$:

$$\forall v \in V,\ \pi_v \circ \alpha = \phi_v \iff \forall c \in C,\ \alpha(c) = \theta(c)$$

Thus, $\Gamma(Q, \mathbf{A}_\bullet)$ is a limit of $(Q, \mathbf{A}_\bullet)$. $\square$

Proposition 1.2 proves that limits of a finite diagram in $\mathrm{Vect}_{\mathrm{Fin-Dim}}$ is the same as the sections of the corresponding quiver with its representation. Proposition 1.1 already gives an algorithm to compute $\Gamma(Q, \mathbf{A}_\bullet)$. However it requires multiple computations in the total space $\prod_{v \in V} \mathbf{A}_v$, which would only be realistic for very small quivers. Section 2 describes an algorithm inspired from [SHN21] to compute the sections space more efficiently.

1.3 Cosections and colimits

Colimits are the dual notion of limits as defined in definition 1.4. Like in the previous paragraph, we want to reformulate colimits in our particular setting by establishing the dual of proposition 1.2. Let $Q = (V, E)$ be a quiver and $\mathbf{A}_\bullet$ a representation of Q.

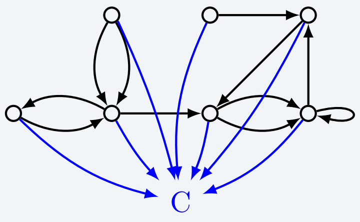
Definition 1.6 (Colimit).

- A cocone of $(Q, \mathbf{A}_\bullet)$ is a finite dimensional vector space C, together with linear maps $\psi_v : \mathbf{A}_v \to C$ for $v \in V$ so that, for all $e \in E$, we have the following compatibility requirement:

$$\psi_{s(e)} = \psi_{t(e)} \circ \mathbf{A}_e$$

- A colimit of $(Q, \mathbf{A}_\bullet)$ is a cocone $(C, (\psi_v)_V)$ so that, for all other cocones $(C', (\psi'_v)_V)$, there is a unique linear map $\alpha : C \to C'$ such that, for all $v \in V$,

$$\psi'_v = \alpha \circ \psi_v$$



$$\begin{tikzcd}
\mathbf{A}_u \arrow[r, "\mathbf{A}_e"] \arrow[dr, "\psi_u"'] \arrow[ddr, bend right, "\psi'_u"'] & \mathbf{A}_v \arrow[d, "\psi_v"] \arrow[dd, bend left, "\psi'_v"] \\
 & C \arrow[d, "\exists!"] \\
 & C'
\end{tikzcd}$$

As for limits, colimits are unique up to isomorphism and are a way to formulate many algebraic constructions.

Example 1.3

Construction	Formula	As colimit of a diagram
Coequalizer	$\mathrm{Coeq}(\mathbf{A}_e, \mathbf{A}_{e'}) :=$ $\mathbf{A}_v \big/ \mathrm{Im}(\mathbf{A}_e - \mathbf{A}_{e'})$	$\begin{tikzcd} u \arrow[r, bend left, "e"] \arrow[r, bend right, "e'"'] & v \end{tikzcd}$
Pushout	$\mathbf{A}_u \otimes_{\mathbf{A}_w} \mathbf{A}_v :=$ $\mathbf{A}_u \bigoplus \mathbf{A}_v \big/ \mathrm{Im}\left((\mathbf{A}_e, -\mathbf{A}_{e'})\right)$	$\begin{tikzcd} & v \\ u & w \arrow[l, "e"] \arrow[u, "e'"'] \end{tikzcd}$

We denote $i_v : \mathbf{A}_v \to \bigoplus_{u\in V} \mathbf{A}_u$ the inclusion maps for $v \in V$. Then, using the map $\omega \begin{cases} \bigoplus_{v\in V} \mathbf{A}_v & \to C \\ \sum_{v\in V} i_v(x_v) & \mapsto \sum_V \psi_v(x_v) \end{cases}$ for every cocone $(C, (\psi_v)_V)$, we will show that one of the isomorphic colimits is a quotient of $\bigoplus_{v\in V} \mathbf{A}_v$ together with the maps induced by the inclusions $(i_v)_V$.

$$\begin{tikzcd} & \bigoplus\limits_{v\in V} \mathbf{A}_v \big/ \ker \omega \arrow[dl, "\omega"'] & \\ C & & \mathbf{A}_v \arrow[ul, "i_v"'] \arrow[ll, "\psi_v"] \end{tikzcd}$$

Definition 1.7 We define the space of cosections

$$\Delta(Q, \mathbf{A}_\bullet) := \bigoplus_{v\in V} \mathbf{A}_v \Big/ \sum_{e\in E} \mathrm{Im}(i_{s(e)} - i_{t(e)} \circ \mathbf{A}_e)$$

where $i_v : \mathbf{A}_v \hookrightarrow \bigoplus_{u\in V} \mathbf{A}_u$ is the canonical inclusion map.

Note that since all spaces have a finite dimension, $\bigoplus_v \mathbf{A}_v = \prod_v \mathbf{A}_v$. We prefer the $\bigoplus$ notation to highlight that this definition is the dual of proposition 1.1. To simplify the expressions, we use the following notation, for any edge $e \in E$ or path p of Q:

$$\mathcal{I}_{\mathbf{A}_\bullet}(e) := \mathrm{Im}\left(i_{s(e)} - i_{t(e)} \circ \mathbf{A}_e\right) \qquad \text{and} \qquad \mathcal{I}_{\mathbf{A}_\bullet}(p) := \mathrm{Im}\left(i_{s(p)} - i_{t(p)} \circ \mathbf{A}_p\right).$$

Example 1.4 Consider the quiver $Q = u \overset{e}{\underset{e'}{\rightrightarrows}} v$

The colimit of a representation $\mathbf{A}_\bullet$ of Q is $\mathrm{Coeq}(\mathbf{A}_e, \mathbf{A}_{e'}) = \mathbf{A}_v \big/ \mathrm{Im}(\mathbf{A}_{e'} - \mathbf{A}_e)$. On the other hand:

$$\begin{aligned} \Delta(Q, \mathbf{A}_\bullet) &= \mathbf{A}_u \bigoplus \mathbf{A}_v \Big/ \mathrm{Im}(i_u - i_v \circ \mathbf{A}_{e'}) + \mathrm{Im}(i_u - i_v \circ \mathbf{A}_e) \\ &= \mathbf{A}_u \bigoplus \mathbf{A}_v \Big/ \mathrm{Im}(i_u - i_v \circ \mathbf{A}_{e'}) \bigoplus \mathrm{Im}(i_v \circ (\mathbf{A}_{e'} - \mathbf{A}_e)) \\ &\cong \mathbf{A}_v \Big/ \mathrm{Im}(\mathbf{A}_{e'} - \mathbf{A}_e) \qquad \text{using corollary } A.1 \end{aligned}$$

One can check that the isomorphism $\Delta(Q, \mathbf{A}_\bullet) \cong \mathrm{Coeq}(\mathbf{A}_e, \mathbf{A}_{e'})$ transforms i_u into $\overline{\mathbf{A}_e}$ and i_v into $\overline{\mathrm{id}_{\mathbf{A}_v}}$ where $\overline{x}$ is the class of $x \in \mathbf{A}_v$ in $\mathbf{A}_v \big/ \mathrm{Im}(\mathbf{A}_{e'} - \mathbf{A}_e)$.

We can prove the dual of proposition 1.2:

Proposition 1.3 $\Delta(Q, \mathbf{A}_\bullet)$, together with the maps induced by the inclusions $(i_v)_{v\in V}$, is a colimit of the Q-shaped diagram corresponding to $\mathbf{A}_\bullet$.

Proof. We denote $\overline{x}$ the class of an element $x \in \bigoplus_{v\in V} \mathbf{A}_v$ in $\Delta(Q, \mathbf{A}_\bullet)$.

For every $e \in E$, the following diagram commutes since for $x \in \mathbf{A}_{s(e)}$,

$$\begin{tikzcd} \mathbf{A}_{s(e)} \arrow[rr, "\mathbf{A}_e"] \arrow[dr, "\overline{i_{s(e)}}"'] & & \mathbf{A}_{t(e)} \arrow[dl, "\overline{i_{t(e)}}"] \\ & \Delta(Q, \mathbf{A}_\bullet) & \end{tikzcd}$$

$$\overline{i_{t(e)}} \circ \mathbf{A}_e(x) = \overline{i_{t(e)} \circ \mathbf{A}_e(x)} = \overline{i_{s(e)}(x)} = \overline{i_{s(e)}}(x)$$

Hence, $\Delta(Q, \mathbf{A}_\bullet)$ is a cocone. Moreover, if $(D, (\psi_v)_{v\in V})$ is another cocone then, for any linear map $\alpha : \Delta(Q, \mathbf{A}_\bullet) \to D$:

$$\begin{aligned} \forall v \in V, \alpha \circ \overline{i_v} = \psi_v \iff & \forall x = (x_v)_{v\in V} \in \bigoplus_{v\in V} \mathbf{A}_v,\ \alpha(\overline{x}) = \sum_{v\in V} \psi_v(x_v) \\ \iff & \alpha \text{ is the quotient map associated with } \sum_{v\in V} \psi_v \circ \pi_v \end{aligned}$$

where $\pi_v : \bigoplus_{u\in V} \mathbf{A}_u \to \mathbf{A}_v$ is the canonical projection. Since D is a cocone, we have for $e \in E$,

$$\begin{aligned} \left(\sum_v \psi_v \circ \pi_v\right) \circ (i_{s(e)} - i_{t(e)} \circ \mathbf{A}_e) &= \psi_{s(e)} \circ \pi_{s(e)} \circ i_{s(e)} - \psi_{t(e)} \circ \pi_{t(e)} \circ i_{t(e)} \circ \mathbf{A}_e \\ &= \psi_{s(e)} - \psi_{t(e)} \circ \mathbf{A}_e = 0. \end{aligned}$$

Hence, $\sum_{v\in V} \psi_v \circ \pi_v$ is trivial on $\sum_{e\in E} \mathrm{Im}(i_{s(e)} - i_{t(e)} \circ \mathbf{A}_e)$. Its quotient map is well defined and it is the only map of cocone from $\Delta(Q, \mathbf{A}_\bullet)$ to D. Finally, $\Delta(Q, \mathbf{A}_\bullet)$ is a colimit of the diagram induced by Q and $\mathbf{A}_\bullet$. $\square$

We fix a basis for each $\mathbf{A}_v$, for $v \in V$. Each $\mathbf{A}_{(u,v)}$ for $(u, v) \in E$ is now represented by a matrix $\dim \mathbf{A}_v \times \dim \mathbf{A}_u$.

Definition 1.8 The *transpose quiver* of Q is $Q^T = (V, E^T := \{(v, u) \mid (u, v) \in E\})$ and the transpose representation of $\mathbf{A}_\bullet$ is $\mathbf{A}_\bullet^T$ defined by $\mathbf{A}_v^T := \mathbf{A}_v$ for $v \in V$ and $\mathbf{A}_{(u,v)}^T := (\mathbf{A}_{(v,u)})^T$ for $(u, v) \in E^T$.

Proposition 1.4 Let $(L, (\phi_v)_{v\in V})$ be a limit of $(Q^T, \mathbf{A}_\bullet^T)$. If we fix a basis of L, then $(L, (\phi_v^T)_v)$ is a colimit of $(Q, \mathbf{A}_\bullet)$.

Proof.

- Let $(u, v) \in E$. Since L is a cone of $(Q^T, \mathbf{A}_\bullet^T)$, one has $\mathbf{A}_{(v,u)}^T \circ \phi_v = \phi_u$ and

$(\phi_v)^T \circ (\mathbf{A}^T_{(v,u)})^T = (\phi_u)^T$ *i.e.* $(\phi_v)^T \circ \mathbf{A}_{(u,v)} = (\phi_u)^T$.

$$\begin{tikzcd}
u & & \\
 & L \arrow[ul, "\phi_u"'] \arrow[dl, "\phi_v"] & cC \arrow[l, dashed, "\alpha"] \arrow[llu, bend right, "(\psi'_v)^T"'] \arrow[lld, bend left, "(\psi'_v)^T"] \\
v \arrow[uu, "\mathbf{A}^T_{(v,u)}"] & &
\end{tikzcd}
\qquad
\begin{tikzcd}
u \arrow[dr, "\phi_u^T"] \arrow[dd, "\mathbf{A}_{(u,v)}"'] \arrow[drr, bend left, "\psi'_u"] & & \\
 & L \arrow[r, dashed, "\alpha^T"] & cC \\
v \arrow[ur, "\phi_v^T"'] \arrow[urr, bend right, "\psi'_v"'] & &
\end{tikzcd}$$

- Let $(cC, (\psi'_v)_{v\in V}))$ be a cocone of $(Q, \mathbf{A}_\bullet)$. Then for $(u, v) \in E$, $(\psi'_u)^T = (\psi'_v \circ \mathbf{A}_{(u,v)})^T = \mathbf{A}^T_{(v,u)} \circ (\psi'_v)^T$ hence $(L, ((\psi'_v)^T)_{v\in V})$ is a cone of $(Q^T, \mathbf{A}^T_\bullet)$. By definition of limits, there is a unique map $\alpha : cC \to L$ such that for all $v \in V$, $(\psi'_v)^T = \phi_v \circ \alpha$ and after transposing this expression, α^T is the only map $L \to cC$ such that $\psi'_v = \alpha^T \circ (\phi_v)^T$ for all $v \in V$. Finally, $(L, (\phi^T_v)_v)$ is a colimit of $(Q, \mathbf{A}_\bullet)$.

□

The colimit of (Q, A) can be computed as the limit of the transpose quiver and representation. A possible method could be to devise an algorithm for limits and to use it on the transpose to compute colimits. This approach would have two drawbacks:

- On the one hand, any vector space of the right dimension could be used as a colimit but the cosections space has a particularly nice definition as a quotient of the total space, which is not given directly by proposition 1.4.
- On the other hand, the method described in this essay could be extended to the computation of (co)limits of finite diagrams in a (co)complete category (see 2.6). In this more general setting, it may not be possible to transpose maps. Worst, the category might not be both complete and cocomplete.

That's why in the following section, computations for sections and cosections are provided as two separate methods.

2 Computing (co)sections

In the last section, we defined (co)limits of finite diagrams of finite-dimensional vector spaces and reformulate it as the space of (co)sections of a quiver representation. We now fix a quiver Q and a representation $\mathbf{A}_\bullet$ of Q. The goal of this section is to design an algorithm to compute the (co)sections spaces $\Gamma(Q, \mathbf{A}_\bullet)$ and $\Delta(Q, \mathbf{A}_\bullet)$. The formulae from proposition 1.1 and definition 1.7 already give possible algorithms. However, those algorithms imply computations in the total space which dimension is too large, except for very small examples.

For larger examples, we describe algorithms that keep computations in a smaller space. The algorithm for sections comes from [SHN21] and I adapted it to the case of cosections by dualizing each step. In this section we follow [SHN21](2-4) while proving in parallel the dual statements.

The main idea of these algorithms is to transform Q and $\mathbf{A}_\bullet$, while keeping the same (co)sections space, until Q is a rooted tree. We will first show that computations are easy for rooted trees (see 2.1). Then we will see how to remove cycles (2.2 for strongly connected quivers and 2.3 for the general case) and finally how to add a root and to remove parallel paths (2.4).

When changing Q and $\mathbf{A}_\bullet$, we want to keep the space of (co)sections unchanged, up to isomorphism. However, in order to retrieve the maps associated with the (co)limit, we need to keep track of these isomorphisms. In fact, the isomorphisms involved in our transformations will mostly be induced by the canonical injection or quotient maps.

Definition 2.1 Let $Q = (V, E)$ and $Q' = (V, E')$ two quivers with their representations $\mathbf{A}_\bullet$ and $\mathbf{A}'_\bullet$. We write

- $(Q, \mathbf{A}_\bullet) \overset{\Gamma}{\triangleleft} (Q', \mathbf{A}'_\bullet)$ if for all $v \in V$, $\mathbf{A}_v \subseteq \mathbf{A}'_v$ and the inclusion $\bigoplus_V \mathbf{A}_v \hookrightarrow \bigoplus_V \mathbf{A}'_v$ induces an isomorphism $\Gamma(Q, \mathbf{A}_\bullet) \cong \Gamma(Q', \mathbf{A}'_\bullet)$.

- $(Q, \mathbf{A}_\bullet) \overset{\Delta}{\triangleleft} (Q', \mathbf{A}'_\bullet)$ if for all $v \in V$, $\mathbf{A}_v = \mathbf{A}'_v \,/\, B_v$ for some $B_v \subseteq \mathbf{A}'_v$ and the quotient map $\bigoplus_V \mathbf{A}'_v \to \bigoplus_V \mathbf{A}_v$ induces an isomorphism $\Delta(Q', \mathbf{A}'_\bullet) \cong \Delta(Q, \mathbf{A}_\bullet)$.

2.1 Case of a tree-like quiver

We start by showing that computing (co)sections of a directed rooted tree is easy. Indeed, when Q is a directed rooted tree, the representation space of the root is either the sections space or the cosections space, depending on whether Q is an out-tree or an in-tree.

2.1.1 Sections of an out-tree

Definition 2.2 Q is called an *out-tree* with root r if, for all v in V, there is a unique path $p[v]$ from r to v.

$$\begin{tikzcd}
 & & r \arrow[dl] \arrow[dr] & \\
 & v_1 \arrow[dl] \arrow[d] \arrow[dr] & & v_2 \\
v_{11} & v_{12} & v_{13} &
\end{tikzcd}$$

Proposition 2.1 If Q is an out-tree with root r then the projection $\pi_r : \prod_{v \in V} \mathbf{A}_v \to \mathbf{A}_r$ induces an isomorphism $\Gamma(Q, \mathbf{A}_\bullet) \cong \mathbf{A}_r$.

Proof. For a path $p = e_1, \ldots, e_m$, we denote $\mathbf{A}_p := \mathbf{A}_{e_m} \circ \cdots \circ \mathbf{A}_{e_1}$. Let's consider the linear map:

$$\Phi \left\{ \begin{array}{ll} \mathbf{A}_r & \to \Gamma(Q, \mathbf{A}_\bullet) \\ x & \mapsto (\mathbf{A}_{p[v]}(x))_{v \in V} \end{array} \right.$$

Φ is well defined. Indeed, for $x \in \mathbf{A}_r$ and $e \in E$, the uniqueness of $p[t(e)]$ justifies the following statement:

$$\mathbf{A}_e(\Phi(x)_{s(e)}) = \mathbf{A}_{p[s(e)],e}(x) = \mathbf{A}_{p[t(e)]}(x) = \Phi(x)_{t(e)}.$$

For $\gamma \in \Gamma(Q, \mathbf{A}_\bullet)$, we have $\Phi \circ \pi_r(\gamma) = \Phi(\gamma_r) = (\mathbf{A}_{p[v]}(\gamma_r))_{v\in V} = \gamma$. Similarly, for $x \in \mathbf{A}_r$, we have:

$$\pi_r \circ \Phi(x) = \pi_r((\mathbf{A}_{p[v]}(x))_v) = x. \qquad \text{Hence, } \Gamma(Q, \mathbf{A}_\bullet) \overset{\pi_r}{\cong} \mathbf{A}_r.$$

□

2.1.2 Cosections of an in-tree

Definition 2.3 Q is called an *in-tree* with root r if, for all $v \in V$, there is a unique path $q[v]$ from v to r.

$$\begin{tikzcd}
 & & r & \\
 & v_1 \arrow[ur] & & v_2 \arrow[ul] \\
v_{11} \arrow[ur] & v_{12} \arrow[u] & v_{13} \arrow[ul] &
\end{tikzcd}$$

Lemma 2.1 Let p be a path in Q. Then, $\quad \mathcal{I}_{\mathbf{A}_\bullet}(p) \subseteq \sum_{i=1}^{m} \mathcal{I}_{\mathbf{A}_\bullet}(e_i)$.

Proof.

$$\begin{aligned}
i_{s(e_1)} - i_{t(e_m)} \circ \mathbf{A}_p &= i_{s(e_1)} - i_{t(e_m)} \circ \mathbf{A}_{e_m} \circ \cdots \circ \mathbf{A}_{e_1} \\
&= \sum_{i=1}^{m} (i_{s(e_i)} - i_{t(e_i)} \circ \mathbf{A}_{e_i}) \circ \mathbf{A}_{e_{i-1}} \circ \cdots \circ \mathbf{A}_{e_1}
\end{aligned}$$

□

Proposition 2.2 If Q is an in-tree with root r, then the inclusion map i_r induces an isomorphism $\mathbf{A}_r \cong \Delta(Q, \mathbf{A}_\bullet)$.

Proof. Let's show first that

$$\sum_{e\in E} \mathcal{I}_{\mathbf{A}_\bullet}(e) = \sum_{v\in V\setminus\{r\}} \mathcal{I}_{\mathbf{A}_\bullet}(q[v]). \tag{1}$$

$\boxed{\subseteq}$ Let $e \in E$. If $t(e) = r$, then $(e) = q[s(e)]$ and $\mathcal{I}_{\mathbf{A}_\bullet}(e) = \mathcal{I}_{\mathbf{A}_\bullet}(q[s(e)])$. Otherwise, by uniqueness of $q[s(e)]$, we have $q[s(e)] = (e, q[t(e)])$ and:

$$\begin{tikzcd}
s(e) \arrow[r, "e"] \arrow[rr, bend right, "{q[s(e)]}"'] & t(e) \arrow[r, "{q[t(e)]}"] & r
\end{tikzcd}$$

$$\begin{aligned}
\mathcal{I}_{\mathbf{A}_\bullet}(e) &= \operatorname{Im}(i_{s(e)} - i_{t(e)} \circ \mathbf{A}_e) \\
&= \operatorname{Im}\left(\left(i_{s(e)} - i_r \circ \mathbf{A}_{q[s(e)]}\right) \; - \; \left(i_{t(e)} - i_r \circ \mathbf{A}_{q[t(e)]}\right) \; \circ \; \mathbf{A}_e\right) \\
&\subseteq \operatorname{Im}\left(i_{s(e)} - i_r \circ \mathbf{A}_{q[s(e)]}\right) + \operatorname{Im}\left(i_{t(e)} - i_r \circ \mathbf{A}_{q[t(e)]}\right) \\
&= \mathcal{I}_{\mathbf{A}_\bullet}(q[s(e)]) + \mathcal{I}_{\mathbf{A}_\bullet}(q[t(e)])
\end{aligned}$$

$\boxed{\supseteq}$ Let $v \in V$. A direct application of lemma 2.1 with $p = q[v] = (e_1, \dots, e_m)$ gives $\mathcal{I}_{\mathbf{A}_\bullet}(q[v]) \subseteq \sum_{i=1}^m \mathcal{I}_{\mathbf{A}_\bullet}(e_i)$.

Using (1), we can now write

$$\Delta(Q, \mathbf{A}_\bullet) = \bigoplus_{v \in V} \mathbf{A}_v \Big/ \sum_{v \neq r} \mathcal{I}_{\mathbf{A}_\bullet}(q[v]).$$

Moreover, $\sum_{v \neq r} \mathcal{I}_{\mathbf{A}_\bullet}(q[v])$ is a direct sum. Indeed for $(x_v)_{v \neq r} \in \prod_{v \neq r} \mathbf{A}_v$, by projecting on $\bigoplus_{v \neq r} \mathbf{A}_v$:

$$\sum_{v \neq r} (i_v - i_r \circ \mathbf{A}_{q[v]})(x_v) = 0 \Longrightarrow \forall v \neq r,\ i_v(x_v) = 0$$
$$\Longrightarrow \forall v \neq r,\ (i_v - i_r \circ \mathbf{A}_{q[v]})(x_v) = 0.$$

Hence,

$$\dim \Delta(Q, \mathbf{A}_\bullet) = \dim \mathbf{A}_r + \sum_{v \neq r} \big(\dim \mathbf{A}_v - \dim\big((i_v - i_r \circ \mathbf{A}_{q[v]})(\mathbf{A}_v)\big)\big) = \dim \mathbf{A}_r.$$

Let $\Psi \begin{cases} \bigoplus_{v \in V} \mathbf{A}_v & \to \mathbf{A}_r \\ \sum_{v \in V} x_v & \mapsto x_r + \sum_{v \neq r} \mathbf{A}_{q[v]}(x_v) \end{cases}$. Since $\Psi \circ i_r = \mathrm{id}_{\mathbf{A}_r}$, Ψ is surjective. Furthermore, for all $v \in V \setminus \{r\}$, $\Psi(\mathcal{I}_{\mathbf{A}_\bullet}(q[v])) = \Psi\left(\mathrm{Im}(i_v - i_r \circ \mathbf{A}_{q[v]})\right) = 0$, so Ψ is also well defined on $\Delta(Q, \mathbf{A}_\bullet)$. Finally, the equality on dimensions implies that the map induced by Ψ on $\Delta(Q, \mathbf{A}_\bullet)$ is an isomorphism which inverse is induced by i_r. □

The results from this subsection allow us to compute (co)sections of an out(in)-tree. We now want to transform a general quiver into a rooted tree. The first step of this process is to remove the cycles.

2.2 Case of a strongly connected quiver

In this paragraph, we study how to remove cycles in a strongly connected quiver. The results from this particular case will be necessary to remove cycles in a general quiver.

Definition 2.4 Q is said to be *strongly connected* if for all $u, v \in V$ there is a path from u to v.

Example 2.1 Cycles and complete graphs are strongly connected quivers.

If Q is strongly connected, we will decompose it as an union of smaller and simpler quivers.

Definition 2.5 A quiver $Q' = (V', E')$ is a *subquiver* of Q if $V' \subseteq V$, $E' \subseteq E$ and for all $e' \in E'$, $(t(e'), s(e')) \in V'^2$. We write $Q' \subseteq Q$.

Remark 2.1
- If we look at Q as a category like in remark 1.1, then a subquiver is a subcategory of Q.
- a path $(e_1, \dots, e_m)$ defines a subquiver Q' with $V' = \{s(e_1), \dots, s(e_m), t(e_m)\}$ and $E' = \{e_1, \dots, e_m\}$. We denote $s(Q') = s(e_1)$ and $t(Q') = t(e_m)$

Definition 2.6 An *ear decomposition* $Q_\bullet$ of Q is a list of $c \in \mathbb{N}^*$ subquivers $\{Q_i = (V_i, E_i) \mid i \in [c]\}$ which are paths of Q such that:

1. $\{E_i \mid i \in [c]\}$ is a partition of E.
2. $s(Q_1) = t(Q_1)$
3. for $i > 1$, $V_i \cap \cup_{j<i} V_j = \{s(Q_i), t(Q_i)\}$

Theorem 2.1 A quiver with at least two vertices is strongly connected if and only if it admits an ear decomposition.

The proof of the theorem is given in [BJG09] through the following algorithm:

Algorithm 1: ear decomposition
```
input: $Q$, a strongly connected quiver
output: $Q_\bullet$ an ear decomposition of $Q$
$Q_\bullet = \{Q_1\}$ where $Q_1$ is any cycle of $Q$;
i=1;
while $\bigcup_{j\leqslant i} V_j \neq V$ do
    Let $e \in E$ such that $s(e) \in \bigcup_{j\leqslant i} V_j$ and $t(e) \notin \bigcup_{j\leqslant i} V_j$;
    Let $p$ be a shortest path from $t(e)$ to $\bigcup_{j\leqslant i} V_j$;
    Add the path $e, p$ to $Q_\bullet$ and increment $i$;
for $e \notin \bigcup_{j\leqslant i} E_i$ do
    Add $e$ to $Q_\bullet$;
```

This algorithm is correct since the strong connectivity of Q implies the existence of Q_1, e and p. Moreover, at the end of the while-loop all the remaining edges are self-loops which are ears on their own.

proof of theorem 2.1
$\boxed{\Rightarrow}$ The result is a consequence of the correctness of algorithm 1.

$\boxed{\Leftarrow}$ By induction on $i \in [c]$, $\displaystyle\bigcup_{j\leqslant i} Q_j := \left(\bigcup_{j\leqslant i} V_j, \bigcup_{j\leqslant i} E_j\right)$ is strongly connected. $\square$

Until the end of this section, we assume Q to be strongly connected, we denote $Q_\bullet$ one of its ear decompositions and we choose $r = s(Q_1) = t(Q_1) \in V_1$.

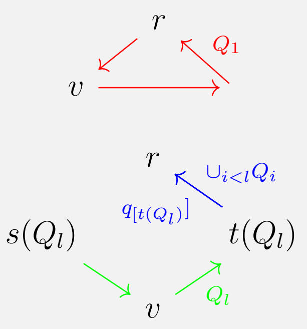
Definition 2.7
- the *depth* $|e|$ of an edge e is the unique $i \in [c]$ such that $e \in E_i$
- A path p is said to be increasing(decreasing) if its edges are in increasing(decreasing) depth order.

- for $v \in V$, $l(v) := \min\{i \in [c] \mid v \in V_i\}$

Proposition 2.3 For any vertex $v \in V \setminus \{r\}$, there exists:

1. a unique $Q_\bullet$-increasing path $p[v]$ from r to v with all edges of depth $\leq l(v)$, and,
2. a unique $Q_\bullet$-decreasing path $q[v]$ from v to r with all edges of depth $\leq l(v)$.

Proof.
We will prove the 2. by induction on $l(v)$.
If $l(v) = 1$, Q_1 is a cycle and there is a unique path with all edges of depth 1 from v to r.
Let $c \geqslant l > 1$. Assume that the property holds for all $u \in \cup_{i<l} V_i$ and let $v \in V$ such that $l(v) = l$. Q_l is a path between $s(Q_l)$ and $t(Q_l)$ so there is a unique path of depth l from v to $\cup_{i<l} V_i$ and it lands in $t(Q_l)$. By induction, there is a unique decreasing path from $t(Q_l)$ to r. Finally, there is a unique decreasing path from v to r.



$\square$

2.2.1 Sections of a strongly connected quiver

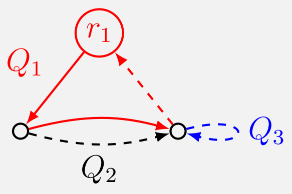
To compute the sections of the strongly connected quiver Q, we will remove the last edge of each ear of $Q_\bullet$ to obtain an out-tree. The representation space of the root will be modified to keep the same space of sections.

More precisely, for each $i \in [c]$, there is a unique $\epsilon_i \in E_i$ called i-th terminal edge, such that $t(\epsilon_i) = t(Q_i)$. We denote $E_{ter} = \{\epsilon_i \mid i \in [c]\}$ the set of terminal edges of Q.

Definition 2.8 The *out-tree induced* by $Q_\bullet$ is the subquiver $T_{out} = T_{out}(Q)$ defined by the vertices V and the edges $E \setminus E_{ter}$.

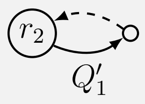
Example 2.2 Let

$R_1 :=$



and

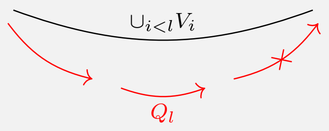
$R_2 :=$



$\{Q_1, Q_2, Q_3\}$ and $\{Q_1{}'\}$ are ear decompositions of respectively R_1 and R_2. The dashed edges are the terminal edges. Thus, $T_{out}(R_1)$ and $T_{out}(R_2)$ are given by the solid lines.

Proposition 2.4 T_{out} is an out-tree with root r.

Proof. Since all paths in T_{out} are increasing, the result is a consequence of proposition 2.3. $\square$



As in section 2.1.1, we denote $p[v]$ the unique path from r to v in T_{out} and we write

$$K(Q_\bullet, \mathbf{A}_\bullet) := \bigcap_{\epsilon \in E_{ter}} \ker\left(\mathbf{A}_{p[t(\epsilon)]} - \mathbf{A}_\epsilon \circ \mathbf{A}_{p[s(\epsilon)]}\right).$$

Proposition 2.5 $\Gamma(Q, \mathbf{A}_\bullet) \cong K(Q_\bullet, \mathbf{A}_\bullet)$.

Proof. By proposition 2.1, $\mathbf{A}_r \overset{\Phi}{\cong} \Gamma(T_{out}, \mathbf{A}_\bullet)$ via $\Phi : \left\{ \begin{array}{ll} \mathbf{A}_r & \to \Gamma(T_{out}, \mathbf{A}_\bullet) \subset \prod_{v \in V} \mathbf{A}_v \\ x & \mapsto (\mathbf{A}_{p[v]}(x))_{v \in V} \end{array} \right.$
Then $\Gamma(Q, \mathbf{A}_\bullet) = \{\gamma \in \Gamma(T_{out}, \mathbf{A}_\bullet) \mid \forall \epsilon \in E_{ter}, \mathbf{A}_\epsilon(\gamma_{s(\epsilon)}) = \gamma_{t(\epsilon)}\}$ and composing by Φ:

$$\Gamma(Q, \mathbf{A}_\bullet) = \Phi\left(\{x \in \mathbf{A}_r \mid \forall \epsilon \in E_{ter}, \mathbf{A}_\epsilon(\mathbf{A}_{p[s(\epsilon)]}(x)) = \mathbf{A}_{p[t(\epsilon)]}(x)\}\right) = \Phi\left(K(Q_\bullet, \mathbf{A}_\bullet)\right)$$

$\square$

Proposition 2.6 Let $\mathbf{A}'_\bullet$ be the representation of T_{out} defined by:

$$\mathbf{A}'_v = \left\{ \begin{array}{ll} \mathbf{A}_v & \text{if } v \neq r \\ K(Q_\bullet, \mathbf{A}_\bullet) & \text{if } v = r \end{array} \right. \quad \text{and} \quad \mathbf{A}'_e = \left\{ \begin{array}{ll} \mathbf{A}_e & \text{if } s(e) \neq r \\ (\mathbf{A}_e)|_{\mathbf{A}'_r} & \text{if } s(e) = r \end{array} \right.$$

where $v \in V$ and $e \in E \setminus E_{ter}$. One has:

$$(T_{out}, \mathbf{A}'_\bullet) \overset{\Gamma}{\lhd} (Q, \mathbf{A}_\bullet)$$

Proof. By propositions 2.1 and 2.5, $\Gamma(T_{out}, \mathbf{A}'_\bullet) \cong K(Q, \mathbf{A}_\bullet) \cong \Gamma(Q, \mathbf{A}_\bullet)$ and the isomorphism is $\Phi|_{K(Q,\mathbf{A})} \circ (\pi_r)|_{\Gamma(T_{out}, \mathbf{A}'_\bullet)}$. By the proof of proposition 2.5, it is induced by the inclusion of the total spaces. $\square$

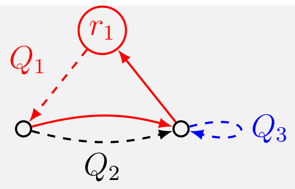
2.2.2 Cosections of a strongly connected quiver

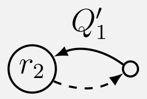
Dually, Q is transformed into a in-tree with the same cosections by removing the first edges of every ear of $Q_\bullet$ and by quotienting the root representation space.

For each $i \in [c]$, there is a unique $\epsilon_i \in E_i$ called i-th initial edge, such that $s(\epsilon_i) = s(Q_i)$. We denote $E_{ini} = \{\epsilon_i \mid i \in [c]\}$ the set of initial edges of Q.

Definition 2.9 The *in-tree induced* by $Q_\bullet$ is the subquiver $T_{in} = T_{in}(Q)$ defined by the vertices V and the edges $E \setminus E_{ini}$.

Example 2.3 Continuing ex. 2.2. $T_{in}(R_1)$ and $T_{in}(R_2)$ are given by the solid lines of:

$R_1 :=$ 

and $R_2 :=$ 

A direct consequence of proposition 2.3 is that:

Proposition 2.7 T_{in} is an in-tree with root r.

As in section 2.1.2, we denote $q[v]$ the unique path from v to r in T_{in} and we write

$$CK(Q_\bullet, \mathbf{A}_\bullet) := {}^{\mathbf{A}_r}\Big/ \sum_{\epsilon \in E_{ini}} \operatorname{Im}(\mathbf{A}_{q[s(\epsilon)]} - \mathbf{A}_{q[t(\epsilon)]} \circ \mathbf{A}_\epsilon)$$

We apply proposition 2.2 to T_{in} and define $\Psi : \bigoplus_{v\in V} \mathbf{A}_v \longrightarrow \mathbf{A}_r$ as in the proof of proposition 2.2.

Lemma 2.2 Ψ induces an isomorphism $\Delta(Q, \mathbf{A}_\bullet) \cong CK(Q_\bullet, \mathbf{A}_\bullet)$.

Proof. Ψ induces a linear map $\tilde{\Psi} : \Delta(Q, \mathbf{A}_\bullet) \to \bigoplus_V \mathbf{A}_v \big/ \Psi(\sum_{e\in E} \mathcal{I}_{\mathbf{A}_\bullet}(e))$ which is surjective because so is Ψ. Since Ψ is null on $\mathcal{I}_{\mathbf{A}_\bullet}(e)$, for every edge e of T_{in}, one has:

$$\begin{aligned}\tilde{\Psi}(\Delta(Q, \mathbf{A}_\bullet)) &= {}^{\mathbf{A}_r}\Big/ \Psi\left(\sum_{e\in E\setminus E_{ini}} \mathcal{I}_{\mathbf{A}_\bullet}(e)\right) + \Psi\left(\sum_{\epsilon\in E_{ini}} \mathcal{I}_{\mathbf{A}_\bullet}(\epsilon)\right) \\ &= {}^{\mathbf{A}_r}\Big/ 0 + \sum_{\epsilon \in E_{ini}} \operatorname{Im} \Psi \circ (i_{s(\epsilon)} - i_{t(\epsilon)} \circ \mathbf{A}_\epsilon)\end{aligned}$$

But from equation (1), $\Psi \circ i_v = \Psi \circ i_r \circ \mathbf{A}_{q[v]} = \mathbf{A}_{q[v]}$ for any $v \in V$. Hence:

$$\tilde{\Psi}(\Delta(Q, \mathbf{A}_\bullet)) = {}^{\mathbf{A}_r}\Big/ \sum_{\epsilon \in E_{ini}} \operatorname{Im}(\mathbf{A}_{q[s(\epsilon)]} - \mathbf{A}_{q[t(\epsilon)]} \circ \mathbf{A}_\epsilon) = CK(Q_\bullet, \mathbf{A}_\bullet).$$

Finally, for $x \in \bigoplus_V \mathbf{A}_v$,

$$\begin{aligned}\tilde{\Psi}\left(x + \sum_{e\in E} \mathcal{I}_{\mathbf{A}_\bullet}(e)\right) = 0 &\iff \Psi(x) \in \Psi\left(\sum_{e\in E} \mathcal{I}_{\mathbf{A}_\bullet}(e)\right) = \Psi\left(\sum_{e\in E_{ini}} \mathcal{I}_{\mathbf{A}_\bullet}(e)\right) \\ &\iff x \in \sum_{e\in E_{ini}} \mathcal{I}_{\mathbf{A}_\bullet}(e) + \ker \Psi = \sum_{e\in E} \mathcal{I}_{\mathbf{A}_\bullet}(e).\end{aligned}$$

Thus, $\tilde{\Psi}$ is injective and $\Delta(Q, \mathbf{A}_\bullet) \overset{\tilde{\Psi}}{\cong} CK(Q_\bullet, \mathbf{A}_\bullet)$. □

Proposition 2.8 Let $\mathbf{A}'_\bullet$ be the representation of T_{in} defined by:

$$\mathbf{A}'_v = \begin{cases} \mathbf{A}_v & \text{if } v \neq r \\ CK(Q_\bullet, \mathbf{A}_\bullet) & \text{if } v = r \end{cases} \qquad \text{and} \qquad \mathbf{A}'_e = \begin{cases} \mathbf{A}_e & \text{if } t(e) \neq r \\ [\mathbf{A}_e] & \text{if } t(e) = r \end{cases}$$

where $[x]$ denotes the class of $x \in \mathbf{A}_r$ in the quotient $CK(Q_\bullet, \mathbf{A}_\bullet)$.

$$(T_{in}, \mathbf{A}'_\bullet) \overset{\Delta}{\triangleleft} (Q, \mathbf{A}_\bullet)$$

Proof. From proposition 2.2 applied to $(T_{in}, \mathbf{A}'_\bullet)$ and lemma 2.2, we have the following ismorphisms

$$\begin{cases} \Delta(T_{in}, \mathbf{A}'_\bullet) & \to CK(Q, \mathbf{A}_\bullet) & \leftarrow \Delta(Q, \mathbf{A}_\bullet) \\ [\ [x]_{\bigoplus_V \mathbf{A}'_v}\ ]_{\Delta(T_{in}, \mathbf{A}'_\bullet)} & \mapsto [\Psi(x)]_{CK(Q,\mathbf{A}_\bullet)} & \leftarrow\!\shortmid [x]_{\Delta(Q,\mathbf{A}_\bullet)} \end{cases} \text{for } x \in \bigoplus_{v \in V} \mathbf{A}_v.$$

So the isomorphism $\Delta(Q, \mathbf{A}_\bullet) \cong \Delta(T_{in}, \mathbf{A}'_\bullet)$ is induced by the quotient map $\bigoplus_{v \in V} \mathbf{A}_v \to \bigoplus_V \mathbf{A}'_v$. □

2.3 Acyclic reduction

Let $Q = (V, E)$ be any quiver with a representation $\mathbf{A}_\bullet$. The goal of this paragraph is to transform $(Q, \mathbf{A}_\bullet)$ into $(Q^*, \mathbf{A}^*_\bullet)$ such that Q^* is acyclic and the space (co)sections of $(Q, \mathbf{A}_\bullet)$ and $(Q^*, \mathbf{A}^*_\bullet)$ are isomorphic via a simple map. The strategy is to apply the previous paragraph, independently, to all maximal strongly connected subquivers of Q, and then to "repair" the representation to obtain a valid representation.

We denote $\mathbf{MSC}(Q)$ the maximal (for the inclusion) strongly connected subquivers of Q. For each subquiver $R \in \mathbf{MSC}(Q)$ we fix an ear decomposition $R_\bullet$ like in definition 2.6.

Proposition 2.9 Distinct subquivers of $\mathbf{MSC}(Q)$ have distinct vertices.

Proof. Assume $w \in R \cap R'$, with $R, R' \in \mathbf{MSC}(Q)$. Then for all $v, v' \in R \cup R'$ there is a path from v to v' going through w. Hence $R \cup R'$ is also strongly connected and R is not maximal. □

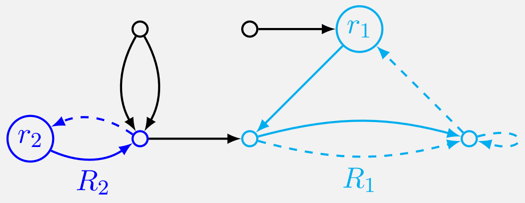
Like previously, we can define for $R \in \mathbf{MSC}(Q)$, the terminal edges of R, $E_{ter}(R)$ (resp. initial edges $E_{ini}(R)$) and the out-tree induced by R, $T_{out}(R)$ (resp. in-tree $T_{in}(R)$). We also denote $\rho(R)$, $V(R)$ and $E(R)$ the root, vertices and edges of $R_\bullet$.

We will define the *acyclic* reduction of Q by removing the terminal (resp. initial edges) of Q. The goal of this subsection is to find a representation of the acyclic reduction with the same (co)sections as Q.

2.3.1 Acyclic reduction conserving sections

Definition 2.10 The acyclic reduction Q^*_{ter} of Q is the subquiver with vertices V and edges $E^* = E \setminus \bigcup_{R \in \mathbf{MSC}(Q)} E_{ter}(R)$.

Example 2.4
Continuing examples 1.1 and 2.2, $\mathbf{MSC}(Q) = \{R_1, R_2\}$. The acyclic reduction (with respect to the ear decomposition from 2.2), Q^*_{ter} is given by the solid lines of:



By construction, Q^*_{ter} as no cycle. In order to build a representation $\mathbf{A}^*_\bullet$ of Q^*_{ter} such that $\Gamma(Q^*_{ter}, \mathbf{A}^*_\bullet) \cong \Gamma(Q, \mathbf{A}_\bullet)$ we start by applying the transformation from last section independently to each $R \in \mathbf{MSC}(Q)$. For $v \in V$ let:

$$\mathbf{A}^{\circ,ter}_v = \begin{cases} K\left(R, (\mathbf{A}_\bullet)_{|R}\right) & \text{if } v = \rho(R) \\ \mathbf{A}_v & \text{else} \end{cases}$$

Note that $\mathbf{A}^{\circ,ter}_\bullet$ may not be a representation of Q^*_{ter}. For instance, in example 2.4, the representation map of the black edge going to r_1 has no reason to land in $K(R_1, (\mathbf{A}_\bullet)_{|R_1})$. Hence, we restrict the representation to account for the dependence between the subquivers of $\mathbf{MSC}(Q)$ and to obtain a valid representation of Q^*_{ter}.

Definition 2.11 Let $R \in \mathbf{MSC}(Q)$ and $v \in V$. The constrained space induced by R on $\mathbf{A}^{\circ,ter}_v$ is:

$$\Lambda_{v,R} := \{x \in \mathbf{A}^{\circ,ter}_v \mid \mathbf{A}_p(x) \in K\left(R, (\mathbf{A}_\bullet)_{|R}\right), \forall p \in \mathcal{P}(Q^*_{ter}, v, \rho(R))\}$$

We define

$$\mathbf{A}^*_v = \bigcap_{R \in \mathbf{MSC}(Q)} \Lambda_{v,R}$$

Proposition 2.10 $\mathbf{A}^*_\bullet$ is a valid representation of Q^*_{ter}.

Proof. Let $e \in E^*$, $R \in \mathbf{MSC}(Q)$ and $x \in \Lambda_{s(e),R}$. For all $p \in \mathcal{P}(Q^*_{ter}, t(e), \rho(R))$, we have, $\mathbf{A}_p(\mathbf{A}_e(x)) = \mathbf{A}_{e,p}(x) \in K(R, (\mathbf{A}_\bullet)_{|R})$ *i.e.* $\mathbf{A}_e(x) \in \Lambda_{t(e),R}$. □

Proposition 2.11

$$(Q^*_{ter}, \mathbf{A}^*_\bullet) \overset{\Gamma}{\lhd} (Q, \mathbf{A}_\bullet)$$

Proof. First, $\bigoplus_v \mathbf{A}^*_v \subseteq \bigoplus_v \mathbf{A}_v$ and $E^* \subsetneq E$ so we only need to prove that $\Gamma(Q, \mathbf{A}_\bullet) \subseteq \bigoplus_v \mathbf{A}^*_v$ and $\Gamma(Q^*_{ter}, \mathbf{A}^*_\bullet) \subseteq \Gamma(Q^*, \mathbf{A}_\bullet)$.

Let $\gamma \in \Gamma(Q, \mathbf{A}_\bullet)$ and $R \in \mathbf{MSC}(Q)$. γ restricts to a section of $\Gamma(R, \mathbf{A}_\bullet)$. Hence, by lemma 2.5, $\gamma_{\rho(R)} \in K\left(R, (\mathbf{A}_\bullet)_{|R}\right) = \mathbf{A}^{\circ,ter}_{\rho(R)}$ and by compatibility, for every $v \in V$ and every $p \in \mathcal{P}(Q^*_{ter}, v, \rho(R))$, $\mathbf{A}_p(\gamma_v) = \gamma_{\rho(R)} \in \mathbf{A}^{\circ,ter}_{\rho(R)}$ i.e. $\gamma_v \in \bigcap_R \Lambda_{v,R} = \mathbf{A}^*_v$.

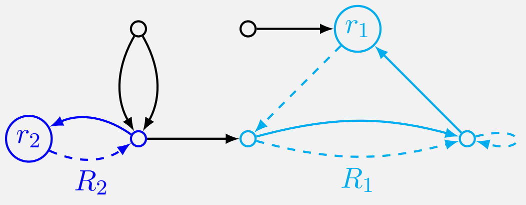
Let $\gamma^* \in \Gamma(Q^*_{ter}, \mathbf{A}^*_\bullet)$. Since $\mathbf{A}^*_e$ is a restriction of $\mathbf{A}_e$ for $e \in E^*$, we only need to check the compatibility on $E \setminus E^*$. Let $\epsilon \in E \setminus E^*$, there is $R \in \mathbf{MSC}(Q)$ such that $\epsilon \in E_{ter}(R)$. If we denote Φ the map from the proof of lemma 2.5 applied to R, we have $(\gamma^*_v)_{v \in V(R)} = \Phi(\gamma^*_{\rho(R)})$ (because of the compatibility on $E \setminus E^*$). Moreover, $\gamma^*_{\rho(R)} \in \mathbf{A}^*_{\rho(R)} \subseteq K\left(R, (\mathbf{A}_\bullet)_{|R}\right)$, so $(\gamma^*_v)_{v \in V(R)} \in \Gamma(R, (\mathbf{A}_\bullet)_{|R})$ and in particular γ^* is $\mathbf{A}_\bullet$-compatible with respect to ϵ. □

2.3.2 Acyclic reduction conserving cosections

Definition 2.12 The acyclic reduction Q^*_{ini} of Q is the subquiver with vertices V and edges $E^* = E \setminus \bigcup_{R \in \mathbf{MSC}(Q)} E_{ini}(R)$.

Example 2.5

Continuing examples 1.1 and 2.3, Q^*_{ini} is given by the solid lines of:



By construction, Q^*_{ini} has no cycle. Our goal is to find a quotient $\mathbf{A}^*_\bullet$ of $\mathbf{A}_\bullet$ such that $\Delta(Q^*_{ini}, \mathbf{A}^*_\bullet) = \Delta(Q, \mathbf{A}_\bullet)$. For $v \in V$ and $R \in \mathbf{MSC}(Q)$ we denote $q^R[v]$ the unique path from v to $\rho(R)$ in $T_{in}(R)$ and we define:

$$I_{p,R} = \sum_{\epsilon \in E_{ini}(R)} \operatorname{Im} \mathbf{A}_p \circ \left(\mathbf{A}_{q^R[s(\epsilon)]} - \mathbf{A}_{q^R[t(\epsilon)]} \circ \mathbf{A}_\epsilon\right)$$

$$\mathbf{A}^*_v = \mathbf{A}_v \Big/ \sum_{\substack{R \in \mathbf{MSC}(Q) \\ p \in \mathcal{P}(Q^*_{ini}, \rho(R), v)}} I_{p,R}$$

For $v = \rho(R)$, we have $\mathcal{P}(Q^*_{ini}, \rho(R), \rho(R)) = \emptyset$. We continue to use the general formula with the convention

$$\sum_{p \in \mathcal{P}(\rho(R), \rho(R))} I_{p,R} = \sum_{e \in E_{ini}(R)} \operatorname{Im}(\mathbf{A}_{q^R[s(e)]} - \mathbf{A}_{q^R[t(e)]} \circ \mathbf{A}_e).$$

Notice that for $R \in \mathbf{MSC}(Q)$, one has $\mathbf{A}_{\rho(R)} \big/ \sum_{p \in \mathcal{P}(\rho(R), \rho(R))} I_{p,R} = CK\left(R, (\mathbf{A}_\bullet)_{|R}\right)$. Quotienting by $\mathcal{P}(Q^*_{ini}, \rho(R), \rho(R))$ is equivalent to applying independently for each $R \in \mathbf{MSC}(Q)$ the transformation from section 2.2. The other paths account for the dependence between the subquivers of $\mathbf{MSC}(Q)$.

Proposition 2.12 For $\epsilon \in E^*$, $\mathbf{A}_\epsilon$ induces a map $\mathbf{A}^*_\epsilon : \mathbf{A}^*_{s(\epsilon)} \to \mathbf{A}^*_{t(\epsilon)}$.

Proof. Let $\epsilon \in E^*$. We will prove that $\mathbf{A}_\epsilon$ sends representatives of a class of $\mathbf{A}^*_{s(\epsilon)}$ to the same class in $\mathbf{A}^*_{t(\epsilon)}$.

- First assume that $s(\epsilon) \neq \rho(R)$ for all $R \in \mathbf{MSC}(Q)$. We only need to check that for $R \in \mathbf{MSC}(Q)$, $e \in E_{ini}(R)$ and $p \in \mathcal{P}(Q^*_{ini}, \rho(R), s(\epsilon))$:

$$\mathbf{A}_\epsilon \left(\operatorname{Im} \mathbf{A}_p \circ (\mathbf{A}_{q^R[s(e)]} - \mathbf{A}_{q^R[t(e)]} \circ \mathbf{A}_e)\right) \subseteq \sum_R \sum_{p \in \mathcal{P}(\rho(R), t(\epsilon))} I_{p,R}.$$

  But $(p, \epsilon) \in \mathcal{P}(Q^*_{ini}, \rho(R), t(\epsilon))$ so

$$\begin{aligned}\mathbf{A}_\epsilon \left(\operatorname{Im} \mathbf{A}_p \circ (\mathbf{A}_{q^R[s(e)]} - \mathbf{A}_{q^R[t(e)]} \circ \mathbf{A}_e)\right) &= \operatorname{Im} \mathbf{A}_{(p,\epsilon)} \circ (\mathbf{A}_{q^R[s(e)]} - \mathbf{A}_{q^R[t(e)]} \circ \mathbf{A}_e) \\ &\subseteq I_{(p,\epsilon),R}\end{aligned}$$

- if $s(\epsilon) = \rho(R)$ for some $R \in \mathbf{MSC}(Q)$, we also need to check that:

$$\mathbf{A}_\epsilon \left(\sum_{e \in E_{ini}(R)} \operatorname{Im}(\mathbf{A}_{q^R[s(e)]} - \mathbf{A}_{q^R[t(e)]} \circ \mathbf{A}_e) \right) \subseteq \sum_R \sum_{p \in \mathcal{P}(\rho(R), t(\epsilon))} I_{p,R}.$$

  Since $(\epsilon) \in \mathcal{P}(Q^*_{ini}, \rho(R), t(\epsilon))$, $\mathbf{A}_\epsilon \left(\sum_{e \in_{ini}(R)} \operatorname{Im}(\mathbf{A}_{q^R[s(e)]} - \mathbf{A}_{q^R[t(e)]} \circ \mathbf{A}_e)\right) = I_{(\epsilon),R}$

$\square$

As a consequence, $\mathbf{A}^*_\bullet$ is a representation of Q^*_{ini}.

Proposition 2.13 $(Q^*_{ini}, \mathbf{A}^*_\bullet) \overset{\Delta}{\triangleleft} (Q, \mathbf{A}_\bullet)$

Proof. By definition:

$$\Delta(Q, \mathbf{A}_\bullet) = \bigoplus_{v \in V} \mathbf{A}_v \Big/ \sum_{e \in E} \mathcal{I}_{\mathbf{A}_\bullet}(e)$$

Let q be the quotient map $\bigoplus_v \mathbf{A}_v \to \bigoplus_v \mathbf{A}^*_v$. By proposition 2.12, $q(\mathcal{I}_{\mathbf{A}_\bullet}(e)) = \mathcal{I}_{\mathbf{A}^*_\bullet}(e)$ for $e \in E^*$, hence q induces an isomorphism:

$$\Delta(Q^*_{ini}, \mathbf{A}^*_\bullet) \cong \bigoplus_{v \in V} \mathbf{A}_v \Big/ \sum_R \sum_{v \in V} i_v \left(\sum_{p \in \mathcal{P}(Q^*_{ini}, \rho(R), v)} I_{p,R} \right) + \sum_{e \in E^*} \mathcal{I}_{\mathbf{A}_\bullet}(e)$$

We need to show that

$$\sum_{e \in E} \mathcal{I}_{\mathbf{A}_\bullet}(e) = \sum_R \sum_{v \in V} i_v \left(\sum_{p \in \mathcal{P}(\rho(R), v)} I_{p,R} \right) + \sum_{e \in E^*} \mathcal{I}_{\mathbf{A}_\bullet}(e)$$

$\boxed{\subseteq}$ Let $\epsilon \in E \backslash E^*$, let's prove that $\mathrm{Im}(i_{s(e)} - i_{t(e)} \circ \mathbf{A}_e)$ is included in the right term. There is a unique $R \in \mathbf{MSC}(Q)$ such that $\epsilon \in E_{ini}(R)$. By lemma 2.1, $\mathrm{Im}(i_{s(\epsilon)} - i_{\rho(R)} \circ \mathbf{A}_{q^R[s(\epsilon)]})$ and $\mathrm{Im}(i_{t(\epsilon)} - i_{\rho(R)} \circ \mathbf{A}_{q^R[t(\epsilon)]})$ are included in $\sum_{e \in E^*} \mathrm{Im}(i_{s(e)} - i_{t(e)} \circ \mathbf{A}_e)$. And:

$$\begin{aligned}
\mathrm{Im}(i_{s(\epsilon)} - i_{t(\epsilon)} \circ \mathbf{A}_\epsilon) \subseteq \mathrm{Im}(i_{s(\epsilon)} - i_{\rho(R)} \circ \mathbf{A}_{q^R[s(\epsilon)]}) & \qquad \subseteq \sum_{e \in E^*} \mathrm{Im}(i_{s(e)} - i_{t(e)} \circ \mathbf{A}_e) \\
+ \mathrm{Im}\, i_{\rho(R)} \circ (\mathbf{A}_{q^R[s(\epsilon)]} - \mathbf{A}_{q^R[t(\epsilon)]} \circ \mathbf{A}_\epsilon) & \qquad = i_{\rho(R)} \left(\sum_{p \in \mathcal{P}(Q^*_{ini}, \rho(R), \rho(R))} I_{p,R} \right) \\
+ \mathrm{Im}(i_{t(\epsilon)} - i_{\rho(R)} \circ \mathbf{A}_{q^R[t(\epsilon)]}) \circ \mathbf{A}_\epsilon & \qquad \subseteq \sum_{e \in E^*} \mathrm{Im}(i_{s(e)} - i_{t(e)} \circ \mathbf{A}_e)
\end{aligned}$$

$\boxed{\supseteq}$ Let $R \in \mathbf{MSC}(Q)$ and $v \in V \setminus \{\rho(R)\}$, let's show that for $p \in \mathcal{P}(Q^*_{ini}, \rho(R), v)$:

$$i_v \left(I_{p,R}\right) \subseteq \sum_{e \in E} \mathrm{Im}(i_{s(e)} - i_{t(e)} \circ \mathbf{A}_e).$$

Let $\epsilon \in E_{ini}(R)$,

$$\begin{aligned}
i_v \circ \mathbf{A}_p \circ \left(\mathbf{A}_{q^R[s(\epsilon)} - \mathbf{A}_{q^R[t(\epsilon)]} \circ \mathbf{A}_\epsilon\right) &= i_v \circ (\mathbf{A}_{(p, q^R[s(\epsilon)])} - \mathbf{A}_{(p, q^R[t(\epsilon)], \epsilon)}) \\
&= \left(i_v \circ \mathbf{A}_{(p, q^R[s(\epsilon)])} - i_{s(\epsilon)}\right) - \left(i_v \circ \mathbf{A}_{(p, q^R[t(\epsilon)], \epsilon)}\right) - i_{s(\epsilon)}\right)
\end{aligned}$$

Then by lemma 2.1 applied to the paths $(p, q^R[t(\epsilon)])$ and $(p, q^R[t(\epsilon)], \epsilon)$ in Q:

$$i_v \circ \mathbf{A}_p \left(\mathrm{Im}\left(\mathbf{A}_{q^R[s(\epsilon)} - \mathbf{A}_{q^R[t(\epsilon)]} \circ \mathbf{A}_\epsilon\right)\right) \subseteq \sum_{e \in E} \mathrm{Im}(i_{s(e)} - i_{t(e)} \circ \mathbf{A}_e).$$

This concludes the proof in the case where $v \neq \rho(R)$. When $v = \rho(R)$, we apply instead the lemma 2.1 to $q^R[t(\epsilon)]$ and $(q^R[t(\epsilon)], \epsilon)$. $\square$

2.4 Arboreal replacement

Let $Q = (V, E)$ be an acyclic quiver and $\mathbf{A}_\bullet$ a representation of Q.

2.4.1 Augmented quiver

The closure of the following relation on Vdefines a partial order on V:

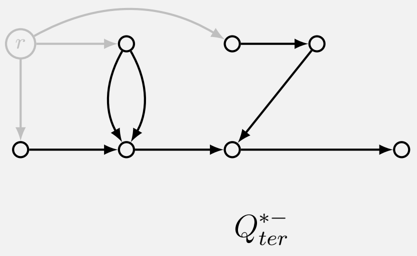
$$u < v \Longleftrightarrow \text{there is a path in } Q \text{ from } u \text{ to } v$$

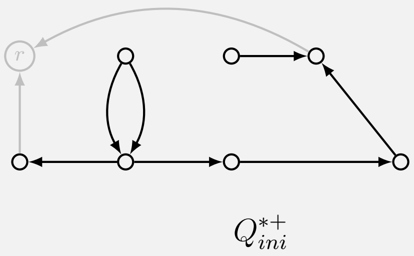
Definition 2.13 We denote V_{min} and V_{max} the minimal and maximal vertices of V for the relation $<$.

- $Q^- := \left(V^- := V \cup \{r\}, E^- := E \cup \bigcup_{v \in V_{min}} \{(r, v)\}\right)$ is the negative augmented quiver

- $Q^+ := \left(V^+ := V \cup \{r\}, E^+ := E \cup \bigcup_{v \in V_{max}} \{(v,r)\}\right)$ is the positive augmented quiver

Example 2.6 Continuing our running example:


Q^{*-}_{ter}


Q^{*+}_{ini}

For $v \in V_{min}$, $\pi_v : \prod_{u \in V_{min}} \mathbf{A}_u \to \mathbf{A}_v$ is the canonical projection and for $v \in V_{max}$, $\iota_v : \mathbf{A}_v \hookrightarrow \bigoplus_{u \in V_{max}} \mathbf{A}_u$ is the inclusion.

Let $\mathbf{A}_\bullet$ be a representation of Q, we define $\mathbf{A}^-_\bullet$ and $\mathbf{A}^+_\bullet$ which are representations of respectively Q^- and Q^+ by:

$$\mathbf{A}^-_v := \begin{cases} \mathbf{A}_v & \text{if } v \in V \\ \prod_{v \in V_{min}} \mathbf{A}_v & \text{if } v = r \end{cases} \quad \text{and} \quad \mathbf{A}^-_e := \begin{cases} \mathbf{A}_e & \text{if } e \in E \\ \pi_v & \text{if } e = (r,v) \text{ with } v \in V_{min} \end{cases}$$

Similarly,

$$\mathbf{A}^+_v := \begin{cases} \mathbf{A}_v & \text{if } v \in V \\ \bigoplus_{v \in V_{max}} \mathbf{A}_v & \text{if } v = r \end{cases} \quad \text{and} \quad \mathbf{A}^+_e := \begin{cases} \mathbf{A}_e & \text{if } e \in E \\ \iota_v & \text{if } e = (v,r) \text{ with } v \in V_{max} \end{cases}$$

Proposition 2.14

- The projection $\bigoplus_{v \in V^-} \mathbf{A}_v \to \bigoplus_{v \in V} \mathbf{A}_v$ induces an isomorphism

$$\Gamma(Q^-, \mathbf{A}^-_\bullet) \cong \Gamma(Q, \mathbf{A}_\bullet)$$

- The inclusion $\bigoplus_{v \in V} \mathbf{A}_v \to \bigoplus_{v \in V^+} \mathbf{A}^+_v$ induces an isomorphism

$$\Delta(Q, \mathbf{A}_\bullet) \cong \Delta(Q^+, \mathbf{A}^+).$$

Proof.

- $\begin{cases} \Gamma(Q, \mathbf{A}_\bullet) & \to \Gamma(Q^-, \mathbf{A}^-_\bullet) \\ \gamma & \mapsto \tilde{\gamma} \\ (\gamma'_v)_{v \in V} & \leftarrow\!\shortmid \gamma' \end{cases}$ where $\tilde{\gamma} := \begin{cases} \gamma_v & \text{if } v \in V \\ (\gamma_u)_{u \in V_{min}} & \text{if } v = r \end{cases}$ is an isomorphism.

- For $v \in V^+$, i^+_v denotes the inclusion $\mathbf{A}^+_v \subseteq \bigoplus_{u \in V^+} \mathbf{A}^+_u$. Since for $v \in V_{max}$, the projection $\bigoplus_{u \in V^+} \mathbf{A}^+_u \to \mathbf{A}^+_v$ induces an isomorphism $\mathcal{I}_{\mathbf{A}^+_\bullet}((v,r)) = \operatorname{Im}(i^+_v - i^+_r \circ \iota_v) \cong \mathbf{A}^+_v$, the following sum is direct:

$$\bigoplus_{v \in V_{max}} \mathcal{I}_{\mathbf{A}^+_\bullet}((v,r)).$$

Moreover, since $\mathbf{A}_r^+ \cap \sum_{e\in E} \mathcal{I}_{\mathbf{A}_\bullet^+}(e) = \{0\}$,

$$\left(\sum_{e\in E} \mathcal{I}_{\mathbf{A}_\bullet^+}(e)\right) \cap \bigoplus_{v\in V_{max}} \mathcal{I}_{\mathbf{A}_\bullet^+}((v,r)) = \{0\}.$$

Hence, using corollary A.1,

$$\begin{aligned}\dim \Delta(Q^+, \mathbf{A}_\bullet^+) &= \dim \Delta(Q, \mathbf{A}_\bullet) + \dim \mathbf{A}_r^+ - \sum_{v\in V_{max}} \dim \operatorname{Im}(i_v^+ - i_r^+ \circ \iota_v) \\ &= \dim \Delta(Q, \mathbf{A}_\bullet) + \sum_{v\in V_{max}} \dim \mathbf{A}_v - \sum_{v\in V_{max}} \dim \mathbf{A}_v \\ &= \dim \Delta(Q, \mathbf{A}_\bullet).\end{aligned}$$

Finally, the inclusion $\bigoplus_{v\in V} \mathbf{A}_v \to \bigoplus_{v\in V^+} \mathbf{A}_v^+$ induces a surjective linear map $\Delta(Q, \mathbf{A}_\bullet) \to \Delta(Q^+, \mathbf{A}_\bullet^+)$ which is also an isomorphism because of the dimensions.

□

Hence we can replace Q by Q^- to compute sections and by Q^+ to compute cosections.

2.4.2 Out-tree replacement

The goal of this paragraph is to compute $\Gamma(Q^-, \mathbf{A}_\bullet^-)$.

Definition 2.14 For $v \in V^-$, we define inductively the *flow space* $\Phi_v \subset \mathbf{A}_r^-$ and the *flow map* $\phi : \Phi_v \to \mathbf{A}_v^-$:

- $\Phi_r := \mathbf{A}_r^-$ and $\phi_r = \mathrm{id}_{\mathbf{A}_r^-}$
- Let $v \neq r$ such that (Φ_u, ϕ_u) is already defined for all $u < v$. In particular, it is defined for the sources of all edges in $E_{in}^-(v) := \{e \in E^- \mid t(e) = v\}$. Then:

  $$\Phi_v := \mathrm{Eq}\left\{\left(\mathbf{A}_e^- \circ \phi_{s(e)}\right)_{|\Phi_v'} \mid e \in E_{in}^-(v)\right\}$$

  and $\phi_v = \left(\mathbf{A}_e^- \circ \phi_{s(e)}\right)_{|\Phi_v}$ for any $e \in E_{in}^-(v)$, where Eq is the equalizer and

  $$\Phi_v' := \bigcap_{e\in E_{in}^-(v)} \Phi_{s(e)}.$$

By construction, the assignement $\left\{\begin{array}{ll}(V^-, \leqslant) & \to (\mathbf{A}_r^-, \subseteq) \\ v & \mapsto \Phi_v\end{array}\right.$ is decreasing. For $v \in V^-$, let $Q_{\leqslant v}^- = (V_{\leqslant v}^-, E_{\leqslant v}^-)$ be the subquiver of Q^- induced by $V_{\leqslant v}^- := \{u \in V^- \mid u \leqslant v\}$.

Proposition 2.15 For $v \in V$ and $p \in \mathcal{P}(Q^-, r, v)$, we have $\phi_v = \left(\mathbf{A}_p^-\right)_{|\Phi_v}$.

Proof. By induction on $(V, <)$ we have:

- For $v \in V_{min}$, $\mathcal{P}(Q^-, r, v) = \{(r,v)\}$, so $\phi_v = (\mathbf{A}_{(r,v)} \circ \phi_r)_{|\Phi_v} = (\mathbf{A}_{(r,v)})_{\Phi_v}$.
- For $v \in V \setminus V_{min}$, any path from r to v is of the form p, e with $p \in \mathcal{P}(Q^-, r, s(e))$ and $e \in E_{in}^-(v)$. Then by induction and since $v \mapsto \Phi_v$ is decreasing:

$$\phi_v = (\mathbf{A}_e \circ \phi_{s(e)})_{|\Phi_v} = (\mathbf{A}_e \circ (\mathbf{A}_p)_{|\Phi_{s(e)}})_{|\Phi_v} = (\mathbf{A}_{p,e})_{|\Phi_v}$$

□

Proposition 2.16 Let $v \in V$, one has $\Phi_v = \bigcap_{u \leqslant v} \operatorname{Eq}\left\{\mathbf{A}_p^- \mid p \in \mathcal{P}(Q^-, r, u)\right\}$.

Proof. $\boxed{\subseteq}$ Let $v \in V$, by proposition 2.15: $\Phi_v \subseteq \operatorname{Eq}\left\{\mathbf{A}_p^- \mid p \in \mathcal{P}(Q^-, r, v)\right\}$ and since $u \mapsto \Phi_u$ is decreasing,

$$\Phi_v \subseteq \bigcap_{u \leqslant v} \Phi_u \subseteq \bigcap_{u \leqslant v} \operatorname{Eq}\left\{\mathbf{A}_p^- \mid p \in \mathcal{P}(Q^-, r, u)\right\}.$$

$\boxed{\supseteq}$ If $v \in V_{min}$, $\Phi_v = \operatorname{Eq}\{\mathbf{A}_{(r,v)}\} = \bigcap_{u \leqslant v} \operatorname{Eq}\left\{\mathbf{A}_p^- \mid p \in \mathcal{P}(Q^-, r, v)\right\}$. Let $v \in V$ such that proposition 2.16 holds for $u \in V_{\leqslant v}^-$, then by induction:

$$\begin{aligned}\bigcap_{u \leqslant v} \operatorname{Eq}\left\{\mathbf{A}_p^- \mid p \in \mathcal{P}(Q^-, r, u)\right\} &\subseteq \bigcap_{e \in E_{in}^-(v)} \bigcap_{u \leqslant s(e)} \operatorname{Eq}\left\{\mathbf{A}_p^- \mid p \in \mathcal{P}(Q^-, r, u)\right\} \\ &\subseteq \bigcap_{e \in E_{in}^-(v)} \Phi_{s(e)} = \Phi'_v\end{aligned}$$

Then using proposition 2.15,

$$\bigcap_{u \leqslant v} \operatorname{Eq}\left\{\mathbf{A}_p^- \mid p \in \mathcal{P}(Q^-, r, u)\right\} \subseteq \operatorname{Eq}\{(\mathbf{A}_{e,p}^-)_{|\Phi'_v} \mid e \in E_{in}^-(v), p \in \mathcal{P}(r, s(e))\} = \Phi_v.$$

□

Proposition 2.17 Let $\gamma \in \prod_{v \in V^-} \mathbf{A}_v^-$, one has for $v \in V$:

$$(\gamma_u)_{u \leqslant v} \in \Gamma(Q_{\leqslant v}^-, \mathbf{A}_\bullet^-) \iff \gamma_r \in \Phi_v \text{ and } \forall u \leqslant v,\ \gamma_u = \phi_u(\gamma_r)$$

Proof. Let $v \in V$,
$\boxed{\Longrightarrow}$ By compatibility, for $u \in V_{\leqslant v}^-$ and $p \in \mathcal{P}(r, u)$, $\mathbf{A}_p(\gamma_r) = \gamma_u$ is independent of p:

$$\gamma_r \in \bigcap_{u \leqslant v} \operatorname{Eq}\left\{\mathbf{A}_p^- \mid p \in \mathcal{P}(Q^-, r, u)\right\} \underset{2.16}{=} \Phi_v \qquad \text{and} \quad \forall u \in V_{\leqslant v}^-,\ \gamma_u \underset{2.15}{=} \phi_u(\gamma_r).$$

$\boxed{\Longleftarrow}$ For $e \in E_{\leqslant v}^-$ there exits $p \in \mathcal{P}(Q^-, r, s(e))$. By proposition 2.15:

$$\mathbf{A}_e(\gamma_{s(e)}) = \mathbf{A}_e \circ \phi_{s(e)}(\gamma_r) = \mathbf{A}_{e,p}(\gamma_r) = \phi_{t(e)}(\gamma_r) = \gamma_{t(e)}.$$

$\square$

Let T^- be any spanning tree of Q^- and $\mathbf{A}'^{\,-}$ the representation of T^- defined by:

$$\mathbf{A}'^{\,-}_v = \begin{cases} \mathbf{A}_v & \text{if } v \neq r \\ \bigcap_{v \in V_{max}} \Phi_v & \text{if } v = r \end{cases} \qquad \text{and} \qquad \mathbf{A}'^{\,-}_e = \begin{cases} \mathbf{A}_e & \text{if } s(e) \neq r \\ (\mathbf{A}_e)|_{\mathbf{A}'^{\,-}_r} & \text{if } s(e) = r \end{cases}$$

Proposition 2.18

$$(T^-, \mathbf{A}'^{\,-}_\bullet) \overset{\Gamma}{\lhd} (Q^-, \mathbf{A}^-_\bullet)$$

Proof. Let $\gamma \in \prod_{v \in V} \mathbf{A}^-_v$, we have:

$$\begin{aligned} \gamma \in \Gamma(Q^-, \mathbf{A}^-_\bullet) &\iff \forall v \in V,\ (\gamma_u)_{u \leqslant v} \in \Gamma(Q^-_{\leqslant v}, \mathbf{A}^-_\bullet) \\ &\iff \forall v \in V, \begin{cases} \gamma_r \in \Phi_v \\ \forall u \leqslant v,\ \gamma_u = \phi_u(\gamma_r) \end{cases} \\ &\iff \begin{cases} \gamma_r \in \bigcap_{v \in V_{max}} \Phi_v \\ \forall v \in V, \gamma_v = \phi_v(\gamma_r) \end{cases} \end{aligned}$$

Hence the map $x \in \bigcap_{v \in V_{max}} \Phi_v \mapsto (\phi_v(x))_{v \in V^-} \in \prod_{V^-} \mathbf{A}_v$ restricts to an isomorphism $\bigcap_{v \in V_{max}} \Phi_v \cong \Gamma(Q^-, \mathbf{A}^-_\bullet)$. But by proposition 2.1, the same map is also an isomorphism $\bigcap_{v \in V_{max}} \Phi_v \cong \Gamma(T^-, \mathbf{A}'^-)$. Thus, $(T^-, \mathbf{A}'^{\,-}_\bullet) \overset{\Gamma}{\lhd} (Q^-, \mathbf{A}^-_\bullet)$. $\square$

2.4.3 In-tree replacement

The goal is to compute $\Delta(Q^+, \mathbf{A}^+)$. Like for out-trees, we introduce for $v \in V$:

- $Q_{\geqslant v} = (V^+_{\geqslant v}, E^+_{\geqslant v})$ the subquiver of Q^+ induced by $\{u \in V^+ \mid u \geqslant v\}$
- $E^+_{out}(v) := \{e \in E^+ \mid s(e) = v\}$ and we choose $e_v \in E^+_{out}(v)$.

The sequence $\begin{cases} u_0 &= v \\ u_{n+1} &= t(e_{u_n}) \end{cases}$ is increasing in $(V^+, \leqslant)$ and stops when $u_n = r$. Since Q^+ is acyclic, it happens in finite time. Hence, we obtain a path $q[v] \in \mathcal{P}(Q^+_{\geqslant v}, v, r)$. Its representation $\mathbf{A}_{q[v]}$ can easily been computed inductively.

Definition 2.15 We define the *co-flow space*, $\Psi_v = \mathbf{A}^+_r \big/ B_v$ where $B_v \subseteq \mathbf{A}^+_r$ is built inductively on $(V^+, \geqslant)$:

- $B_r := 0$
- assume B_u is defined for $u > v$, then let $B'_v := \sum_{e \in E^+_{out}(v)} B_{t(e)}$ and

$$B_v := B'_v + \sum_{(e,e') \in E^+_{out}(v)^2} \operatorname{Im}\left(\mathbf{A}_{q[t(e)]} \circ \mathbf{A}_e - \mathbf{A}_{q[t(e')]} \circ \mathbf{A}_{e'}\right).$$

From the definition, it is clear that $\left\{ \begin{array}{ll} (V^+, \geqslant) & \to (\text{subspaces of } \mathbf{A}_r^+, \subseteq) \\ v & \mapsto B_v \end{array} \right.$ is increasing and thus $v \mapsto \dim \Psi_v$ is decreasing. The next proposition shows that for $v \in V$, $[\mathbf{A}^+_{q[v]}]_{\Psi_v}$, the class of $\mathbf{A}^+_{q[v]}$ in Ψ_v plays the role of ϕ_v in paragraph 2.4.2.

Proposition 2.19 Let $v \in V$, then for all $p \in \mathcal{P}(Q^+, v, r)$, one has $[\mathbf{A}^+_p]_{\Psi_v} = [\mathbf{A}^+_{q[v]}]_{\Psi_v}$.

Proof. For $v = r$, $\mathcal{P}(Q^+, r, r) = \emptyset$. Let $v \in V$ such that the property holds for all $u > v$. Let $p \in \mathcal{P}(Q^+, v, r)$, p can be decomposed into $p = e, p_0$ where $e \in E^+_{out}(v)$ and $p_0 \in \mathcal{P}(Q^+, t(e), r)$. We need to prove that $\mathrm{Im}(\mathbf{A}^+_p - \mathbf{A}^+_{q[v]}) \subseteq B_v$. By induction, $\mathrm{Im}(\mathbf{A}^+_{p_0} - \mathbf{A}^+_{q[t(e)]}) \subseteq B_{t(e)}$ and

$$\begin{aligned} \mathrm{Im}\left(\mathbf{A}^+_p - \mathbf{A}^+_{q[v]}\right) &= \mathrm{Im}\left(\mathbf{A}^+_{p_0} \circ \mathbf{A}^+_e - \mathbf{A}^+_{q[t(e_v)]} \circ \mathbf{A}^+_{e_v}\right) \\ &\subseteq \mathrm{Im}\left((\mathbf{A}^+_{p_0} - \mathbf{A}^+_{q[t(e)]}) \circ \mathbf{A}^+_e\right) + \mathrm{Im}(\mathbf{A}^+_{q[t(e)]} \circ \mathbf{A}^+_e - \mathbf{A}^+_{q[t(e_v)]} \circ \mathbf{A}^+_{e_v}) \\ &\subseteq B_{t(e)} + B_v = B_v \end{aligned}$$

□

Proposition 2.20 For $v \in V$,

$$B_v = \sum_{u \geqslant v} \sum_{\mathcal{P}(u,r)^2} \mathrm{Im}(\mathbf{A}^+_p - \mathbf{A}^+_{p'})$$

where the second sum is over $(p, p') \in \mathcal{P}(Q^+, u, r)^2$.

Proof. The result is proved by induction.

- For $v = r$, $B_r = 0$ and the second sum is empty.
- if the equality is valid for $u > v$, since $u \mapsto B_u$ is increasing:

$$\sum_{u \geqslant v} \sum_{\mathcal{P}(u,r)^2} \mathrm{Im}(\mathbf{A}^+_p - \mathbf{A}^+_{p'}) = \underbrace{\sum_{u > v} B_u}_{=B'_v} + \sum_{p,p' \in \mathcal{P}(v,r)} \mathrm{Im}(\mathbf{A}^+_p - \mathbf{A}^+_{p'})$$

  The result to prove becomes:

$$\underbrace{B'_v + \sum_{e,e' \in E^+_{out}(v)} \mathrm{Im}(\mathbf{A}^+_{q[t(e)]} \circ \mathbf{A}^+_e - \mathbf{A}^+_{q[t(e')]} \circ \mathbf{A}^+_{e'})}_{\Psi_v} = B'_v + \sum_{p,p' \in \mathcal{P}(v,r)} \mathrm{Im}(\mathbf{A}^+_p - \mathbf{A}^+_{p'}).$$

$\boxed{\subseteq}$ Let $e, e' \in E^+_{out}(v)$, since $e, q[t(e)]$ and $e', q[t(e')]$ are paths of $\mathcal{P}(Q^+, v, r)$,

$$\mathrm{Im}(\mathbf{A}^+_{q[t(e)]} \circ \mathbf{A}^+_e - \mathbf{A}^+_{q[t(e')]} \circ \mathbf{A}^+_{e'}) = \mathrm{Im}(\mathbf{A}^+_{e,q[t(e)]} - \mathbf{A}^+_{e',q[t(e')]}) \subseteq \sum_{\mathcal{P}(v,r)^2} \mathrm{Im}(\mathbf{A}^+_p - \mathbf{A}^+_{p'}).$$

$\boxed{\supseteq}$ Let $(p, p') \in \mathcal{P}(Q^+, v, r)^2$, by proposition 2.19, $[\mathbf{A}^+_p]_{\Psi_v} = [\mathbf{A}^+_{q[v]}]_{\Psi_v} = [\mathbf{A}^+_{p'}]_{\Psi_v}$, so $\mathrm{Im}(\mathbf{A}^+_p - \mathbf{A}^+_{p'}) \subseteq \Psi_v$.

$\square$

Lemma 2.3 Let $v \in V$, one has

$$\sum_{e \in E^+_{\geqslant v}} \mathcal{I}_{\mathbf{A}^+_\bullet}(e) = \sum_{u \geqslant v} \mathcal{I}_{\mathbf{A}^+_\bullet}(e_u) + i^+_r(B_v).$$

Proof.
$\boxed{\subseteq}$ Let $e \in E^+_{\geqslant v}$,

$$\begin{aligned} i^+_{s(e)} - i^+_{t(e)} \circ \mathbf{A}^+_e = & \left(i^+_{s(e)} - i^+_r \circ \mathbf{A}^+_{q[s(e)]}\right) \\ & - \left(i^+_{t(e)} - i^+_r \circ \mathbf{A}^+_{q[t(e)]}\right) \circ \mathbf{A}^+_e \\ & + i^+_r \circ \left(\mathbf{A}^+_{q[s(e)]} - \mathbf{A}^+_{q[t(e)]} \circ \mathbf{A}^+_e\right) \end{aligned}$$

$$\begin{tikzcd} & r & \\ s(e) \arrow[ur, "{q[s(e)]}"] \arrow[rr, "e"] & & t(e) \arrow[ul, "{q[t(e)]}"'] \end{tikzcd}$$

So,

$$\mathcal{I}_{\mathbf{A}^+_\bullet}(e) \subseteq \mathcal{I}_{\mathbf{A}^+_\bullet}(q[s(e)]) + \mathcal{I}_{\mathbf{A}^+_\bullet}(q[t(e)]) + i^+_r \left(\mathrm{Im}(\mathbf{A}^+_{q[s(e)]} - \mathbf{A}^+_{e,q[t(e)]})\right).$$

By lemma 2.1, for $u' \leqslant v$, $\mathcal{I}_{\mathbf{A}^+_\bullet}(q[u']) \subseteq \sum_{u \leqslant v} \mathcal{I}_{\mathbf{A}^+_\bullet}(e_u)$. Moreover, by proposition 2.20, $\mathrm{Im}(\mathbf{A}^+_{q[s(e)]} - \mathbf{A}^+_{e,q[t(e)]}) \subseteq B_v$. Hence, $\mathcal{I}_{\mathbf{A}^+_\bullet}(e) = \sum_{u \geqslant v} \mathcal{I}_{\mathbf{A}^+_\bullet}(e_u) + i^+_r(B_v)$.
$\boxed{\supseteq}$ Using prop. 2.20, we only need to prove that for $u \in V^+_{\geqslant v}$ and $p, p' \in \mathcal{P}(Q^+, u, r)$,

$$\mathrm{Im}\left(i^+_r \circ \mathbf{A}^+_p - i^+_r \circ \mathbf{A}^+_{p'}\right) \subseteq \sum_{e \in E^+_{\geqslant v}} \mathcal{I}_{\mathbf{A}^+_\bullet}(e).$$

For such u, p, p', one has

$$\mathrm{Im}\left(i^+_r \circ \mathbf{A}^+_p - i^+_r \circ \mathbf{A}^+_{p'}\right) = \mathrm{Im}\left(\left(i^+_u - i^+_r \circ \mathbf{A}^+_{p'}\right) - \left(i^+_u - i^+_r \circ \mathbf{A}^+_p\right)\right) \subseteq \mathcal{I}_{\mathbf{A}^+_\bullet}(p) + \mathcal{I}_{\mathbf{A}^+_\bullet}(p')$$

But lemma 2.1 implies that $\mathcal{I}_{\mathbf{A}^+_\bullet}(p)$ and $\mathcal{I}_{\mathbf{A}^+_\bullet}(p')$ are included in $\sum_{E^+_{\geqslant v}} \mathcal{I}_{\mathbf{A}^+_\bullet}(e)$. Finally, $i^+_r(B_v) = \sum_{u,p,p'} i^+_r \left(\mathrm{Im}(\mathbf{A}^+_p - \mathbf{A}^+_{p'})\right) \subseteq \sum_{E^+_{\geqslant v}} \mathcal{I}_{\mathbf{A}^+_\bullet}(e)$. $\square$

Lemma 2.4 Let $W \subseteq V$. The following sum is direct:

$$\left(\bigoplus_{u \in W} \mathcal{I}_{\mathbf{A}_\bullet^+}(e_u)\right) \bigoplus i_r^+(\mathbf{A}_r^+).$$

Proof. We prove the result by induction on $\mathrm{card}(W)$.

- For $W = \{v\}$, if $v \notin V_{max}$, then it is obvious. Else if $v \in V_{max}$, let $y \in \mathbf{A}_v^+$ such that $i_v^+(y) - i_r^+ \circ \mathbf{A}_{e_v}^+(y) \in \mathrm{Im}(i_r^+)$. We have $i_v^+(y) = 0$ so $i_v^+(y) - i_r^+(\mathbf{A}_{e_v}^+)(y) = 0$ and $\mathcal{I}_{\mathbf{A}_\bullet^+}(e_v) \cap \mathrm{Im}(i_r^+) = \{0\}$.
- Let $W \subseteq V$ such that the result holds for any W' with $\mathrm{card}\, W' < \mathrm{card}\, W$. Assume

$$0 = \sum_{r > u \geqslant v} (i_u^+ - i_{t(e_u)} \circ \mathbf{A}_{e_u})(x_u) + i_r^+(x_r)$$

where $x_u \in \mathbf{A}_u$ for $u \in W$ and $x_r \in \mathbf{A}_v^+$. Let $v \in \min W$, then u is not the target of any of the $(e_u)_{u \in W}$. Hence, the projection on $\mathbf{A}_v^+$ gives $i_v^+(x_v) = 0$ and $x_v = 0$. Applying the induction hypothesis with $W \setminus \{v\}$, we obtain that for all $u \in W \cup \{r\}$, $x_u = 0$ *i.e.* the sum is direct.

$\square$

Proposition 2.21 For $v \in V$, the map $\Psi \begin{cases} \bigoplus_{u \geqslant v} \mathbf{A}_u^+ & \to \mathbf{A}_r^+ \\ (x_u)_{u \geqslant v} & \mapsto \sum_{u \geqslant v} \mathbf{A}_{q[u]}^+(x_u) \end{cases}$ (where $\mathbf{A}_{q[r]}^+ := \mathrm{id}_{\mathbf{A}_r^+}$) induces an isomorphism $\Delta(Q_{\geqslant v}, \mathbf{A}_\bullet^+) \cong \Psi_v$.

Proof. By using lemma 2.3,

$$\Delta(Q_{\geqslant v}, \mathbf{A}_\bullet^+) = \bigoplus_{u \geqslant v} \mathbf{A}_u^+ \Big/ \sum_{e \in E_{\geqslant v}^+} \mathcal{I}_{\mathbf{A}_\bullet^+}(e) = \bigoplus_{u \geqslant v} \mathbf{A}_u^+ \Big/ \sum_{u \geqslant v} \mathcal{I}_{\mathbf{A}_\bullet^+}(e_u) + i_r^+(B_v)$$

By lemma 2.4 applied to $V_{\geqslant v}$, the following sum is direct:

$$\left(\bigoplus_{u \geqslant v} \mathcal{I}_{\mathbf{A}_\bullet^+}(e_u)\right) \bigoplus i_r^+(B_v).$$

Hence,

$$\dim \Delta(Q_{\geqslant v}, \mathbf{A}_\bullet^+) = \sum_{r > u \geqslant v} (\dim \mathbf{A}_u^+ - \dim \mathcal{I}_{\mathbf{A}_\bullet^+}(e_u)) + \dim \mathbf{A}_r^+ - \dim B_v = \dim \mathbf{A}_r^+ \Big/ B_v$$

Finally, $\Delta(Q_{\geqslant v}, \mathbf{A}_\bullet^+) \cong \Psi_v$.

Ψ is well defined from $\Delta(Q_{\geqslant v}, \mathbf{A}^+_\bullet)$ to Ψ_v since $\Psi(i^+_r(B_v)) = B_v$ and for $r > u \geqslant v$ and $x \in \mathbf{A}^+_u$, $\Psi(i^+_u(x) - i_{t(e_u)} \circ \mathbf{A}^+_{e_u}(x)) = \mathbf{A}_{q[u]}(x) - \mathbf{A}_{e_u,q[t(e_u)]}(x) \in B_v$. Moreover, Ψ is injective. By the equality of dimensions, Ψ induces the desired ismorphism $\Delta(Q_{\geqslant v}, \mathbf{A}^+_\bullet) \cong \Psi_v$. □

Let T^+ be a spanning tree of Q^+ and $\mathbf{A}'^+$ the representation of T^+ defined by

$$\mathbf{A}'^+_v = \begin{cases} \mathbf{A}_v & \text{if } v \neq r \\ \mathbf{A}^+_r \,/\, \sum_{v \in V_{min}} B_v & \text{if } v = r \end{cases} \quad \text{and} \quad \mathbf{A}'^+_e = \begin{cases} \mathbf{A}_e & \text{if } t(e) \neq r \\ [\mathbf{A}_e]_{\sum_{V_{min}} B_v} & \text{if } t(e) = r \end{cases}.$$

Proposition 2.22

$$(T^+, \mathbf{A}'^+_\bullet) \overset{\Delta}{\triangleleft} (Q^+, \mathbf{A}^+_\bullet).$$

Proof. Let $\Psi \begin{cases} \bigoplus_{u \in V^+} \mathbf{A}^+_u & \to \mathbf{A}^+_r \\ (x_u)_{u \in V^+} & \mapsto \sum_{u \in V^+} \mathbf{A}^+_{q[u]}(x_u) \end{cases}$. For the same reason as in the proof of proposition 2.21, Ψ is well defined from $\Delta(Q, \mathbf{A}^+_\bullet)$ to $\mathbf{A}^+_r \,/\, \sum_{v \in V_{min}} B_v$ and is injective. Moreover, for $y \in \mathbf{A}^+_r$ and $v \in V_{min}$, by the isomorphism of proposition 2.21, there is $x \in \bigoplus_{V^+} \mathbf{A}^+_u$ with $\Psi(x) \in y + B_v \subseteq y + \sum_{v \in V_{min}} B_v$. Hence Ψ induces an isomorphism $\Delta(Q^+, \mathbf{A}^+_\bullet) \cong \mathbf{A}^+_r \,/\, \sum_{v \in V_{min}} B_v$ and since for $x \in \bigoplus_{u \in V^+} \mathbf{A}^+_u$ $[(\Psi(x), 0, \ldots, 0)]_{\Delta(Q^+, \mathbf{A}^+_\bullet)} = [x]_{\Delta(Q^+, \mathbf{A}^+_\bullet)}$, the inverse of this isomorphism is induced by i^+_r.

By the proof of proposition 2.2 applied to $(T^+, \mathbf{A}'^+_\bullet)$, Ψ also induces an isomorphism $\Delta(T^+, \mathbf{A}'^+_\bullet) \cong \mathbf{A}^+_r \,/\, \sum_{v \in V_{min}} B_v$, which inverse is induced by the inclusion of $\mathbf{A}^+_r \,/\, \sum_{v \in V_{min}} B_v$ in $\bigoplus_u \mathbf{A}'^+_u$. Hence $\begin{cases} \Delta(Q^+, \mathbf{A}^+_\bullet) & \to \Delta(T^+, \mathbf{A}^+_\bullet) \\ [x]_{\Delta(Q^+, \mathbf{A}^+_\bullet)} & \mapsto [(\Psi(x), 0, \ldots, 0)]_{\Delta(T^+, \mathbf{A}^+_\bullet)} \end{cases}$ induces an isomorphism $\Delta(Q_{\geqslant v}, \mathbf{A}^+_\bullet) \cong \Delta(T^+, \mathbf{A}'^+_\bullet)$. But for $x \in \bigoplus_{V^+} \mathbf{A}^+_u$ and denoting $q'[u]$ the path from u to r in T^+:

$$\begin{aligned} [(\Psi(x), 0, \ldots, 0)]_{\Delta(T^+, \mathbf{A}'^+_\bullet)} &= [(x_r + \sum_{u \neq r} \mathbf{A}'^+_{q[u]}(x_u), 0, \ldots, 0)]_{\Delta(T^+, \mathbf{A}'^+_\bullet)} && \text{by definition} \\ &= [(x_r + \sum_{u \neq r} \mathbf{A}'^+_{q'[u]}(x_u), 0, \ldots, 0)]_{\Delta(T^+, \mathbf{A}'^+_\bullet)} && \text{by prop 2.19} \\ &= [(x_r, (x_u)_{u \neq r})]_{\Delta(T^+, \mathbf{A}^+_\bullet)} && \text{since } \mathcal{I}_{\mathbf{A}'^+_\bullet}(q') = 0 \text{ in } \Delta(T^+, \mathbf{A}'^+_\bullet) \end{aligned}$$

Finally, $(T^+, \mathbf{A}'^+_\bullet) \overset{\Delta}{\triangleleft} (Q^+, \mathbf{A}^+_\bullet)$. □

2.5 Connected components

A simpler trick to improve efficiency of (co)sections computations is to process separately each connected component. This technique is independent, but not incompatible, with the method described above. It is not part of [SHN21], and is my own idea because I thought that isolated vertices may sometimes arise in real applications.

Let $Q = (V, E)$ be a quiver with a representation $\mathbf{A}_\bullet$. Assume that V can be partitioned into two subsets $V_1 \sqcup V_2 = V$ without any edge between the two: there are two subquivers $Q_1 = (V_1, E_1)$ and $Q_2 = (V_2, E_2)$ such that $E = E_1 \sqcup E_2$. Let $\mathbf{A}^1_\bullet$ and $\mathbf{A}^2_\bullet$ be the associated restrictions of $\mathbf{A}_\bullet$ to Q_1 and Q_2.

Proposition 2.23

$$\Gamma(Q, \mathbf{A}_\bullet) = \Gamma(Q_1, \mathbf{A}^1_\bullet) \times \Gamma(Q_2, \mathbf{A}^2_\bullet)$$

Proof. For $v \in V$, we denote the projections $\pi_v : \prod_{u \in V} \mathbf{A}_u \to \mathbf{A}_v$ and for $i \in \{1, 2\}$ such that $v \in V_i$, $\pi^i_v : \prod_{u \in V_i} \mathbf{A}_u \to \mathbf{A}_v$.

$$\begin{aligned}\Gamma(Q, \mathbf{A}_\bullet) &= \Big\{(u_1, u_2) \in \big(\prod_{V_1} \mathbf{A}_v\big) \times \big(\prod_{V_2} \mathbf{A}_v\big) \;\Big|\; \forall e \in E_1 \sqcup E_2, \pi_{t(e)}((u_1, u_2)) = \\ &\qquad \mathbf{A}_e \circ \pi_{s(e)}((u_1, u_2))\Big\} \\ &= \Big\{(u_1, u_2) \in \big(\prod_{V_1} \mathbf{A}_v\big) \times \big(\prod_{V_2} \mathbf{A}_v\big) \;\Big|\; \begin{array}{ll} \forall e \in E_1, & \pi^1_{t(e)}(u_1) = \mathbf{A}_e \circ \pi^1_{s(e)}(u_1) \\ \forall e \in E_2, & \pi^2_{t(e)}(u_2) = \mathbf{A}_e \circ \pi^2_{s(e)}(u_2) \end{array}\Big\} \\ &= \Gamma(Q_1, \mathbf{A}^1_\bullet) \times \Gamma(Q_2, \mathbf{A}^2_\bullet)\end{aligned}$$

□

More precisely $\Gamma(Q, \mathbf{A}_\bullet) = \Gamma(Q_1, \mathbf{A}^1_\bullet) \times \Gamma(Q_2, \mathbf{A}^2_\bullet)$ is induced by the isomorphism $\big(\prod_{V_1} \mathbf{A}_v\big) \times \big(\prod_{V_2} \mathbf{A}_v\big) \cong \prod_V \mathbf{A}_v$.

Proposition 2.24

$$\Delta(Q, \mathbf{A}_\bullet) = \Delta(Q_1, \mathbf{A}^1_\bullet) \oplus \Delta(Q_2, \mathbf{A}^2_\bullet)$$

Proof.

$$\begin{aligned}\Delta(Q, \mathbf{A}_\bullet) &= \big(\bigoplus_{V_1} \mathbf{A}_v\big) \oplus \big(\bigoplus_{V_2} \mathbf{A}_v\big) \Big/ \big(\bigoplus_{E_1} \mathcal{I}_{\mathbf{A}_\bullet}(e)\big) \oplus \big(\bigoplus_{E_2} \mathcal{I}_{\mathbf{A}_\bullet}(e)\big) \\ &= \bigoplus_{V_1} \mathbf{A}_v \Big/ \bigoplus_{E_1} \mathcal{I}_{\mathbf{A}_\bullet}(e) \oplus \bigoplus_{V_2} \mathbf{A}_v \Big/ \bigoplus_{E_2} \mathcal{I}_{\mathbf{A}_\bullet}(e) \\ &= \Delta(Q_1, \mathbf{A}^1_\bullet) \oplus \Delta(Q_2, \mathbf{A}^2_\bullet)\end{aligned}$$

□

More precisely $\Delta(Q, \mathbf{A}_\bullet) = \Delta(Q_1, \mathbf{A}^1_\bullet) \oplus \Delta(Q_2, \mathbf{A}^2_\bullet)$ is induced by the isomorphism $\big(\bigoplus_{V_1} \mathbf{A}_v\big) \oplus \big(\bigoplus_{V_2} \mathbf{A}_v\big) \cong \bigoplus_V \mathbf{A}_v$.

2.6 Generalisation to other categories

In remark 1.1, a quiver with a representation is seen as a finite diagram in the category $\mathrm{Vect}_{\mathrm{Fin-dim}}(\mathbb{F})$ of finite dimensional vector spaces over a field $\mathbb{F}$. Annex A.2 defines the notion of finite diagram in a general category. The method described in this section to compute (co)limits for quivers with a representation could be extended to diagrams in other categories. As an example, we give here a generalised proof of the computation of limits in out-trees and strongly-connected quivers.

Let $\mathcal{C}$ be a category with all products and equalizers. By proposition A.2, finite limits exist in this category. We use the notations of annex A.2, for instance denoting $\times$ the pullback. Let $D : \mathcal{J} \to \mathcal{C}$ be a finite diagram in $\mathcal{C}$. We name the objects of $\mathcal{J}$, $\operatorname{ob} \mathcal{J} = \{A_1, A_2, \ldots, A_n\}$. [SHN21] suggested but did not prove that a generalisation of the method was possible. The following propositions and proofs are my own.

Proposition 2.25 Assume D to be an out-tree i.e. $\operatorname{card}\hom_{\mathcal{J}}(A_i, A_j) = \mathbb{1}_{i \leqslant j}$ for $1 \leqslant i, j \leqslant n$. Then denoting A_{ij} the unique map of $\hom_{\mathcal{J}}(A_i, A_j)$, we have that $(D(A_1), D(A_{1v})_v)$ is a limit of D.

Proof.
First, it is a cone of D since for $1 \leqslant i < j \leqslant n$, $A_{ij} \circ A_{1i} = A_{1j}$ by uniqueness of A_{1j}. Then if $(C, (\phi_v)_v$ is another cone of D, and $\alpha : C \to D(A_1)$,

$$\forall v,\ \phi_v = A_{1v} \circ \alpha \iff \alpha = \phi_1$$

So $(D(A_1), D(A_{1v})_v)$ is a limit of D.

$$\begin{tikzcd}
& C \arrow[d, "\phi_1"] \arrow[ddl, bend right, "\phi_2"'] \arrow[ddr, bend left, "\phi_3"] & \\
& A_1 \arrow[dl, "A_{12}"'] \arrow[dr, "A_{13}"] & \\
A_2 & & A_3
\end{tikzcd}$$

$\square$

Proposition 2.26 Assume D to be a strongly connected diagram (for $1 \leqslant i, j \leqslant n$, $\operatorname{card}(\hom_{\mathcal{J}}(A_i, A_j)) > 0$). Let $Q_\bullet$ be an ear decomposition of the associated quiver Q with root r. By removing all the maps of $\mathcal{J}$ corresponding to terminal edges of $Q_\bullet$, we obtain an out-tree diagram $D' : \mathcal{J}' \to \mathcal{C}$. We denote $A_{i,j}$ the unique map of $\hom_{\mathcal{J}'}(A_i, A_j)$. Let

$$L = \underset{D(A_r)}{\times} \left(\operatorname{Eq}\left(D(A_{1,t(\epsilon)}), D(A_{s(\epsilon),t(\epsilon)}) \circ D(A_{1,s(\epsilon)})\right)\right)_{\epsilon \in E_{ter}(Q)}$$

and let $\phi : L \to D(A_r)$ be the map defined by the pullbacks. Together with the maps $(D(A_{1,v}) \circ \phi)_v$, L is a limit of D.

Proof. For $\epsilon \in E_{ter}(Q)$ we denote $B_\epsilon := \operatorname{Eq}\left(A_{1,t(\epsilon)}, A_{s(\epsilon),t(\epsilon)} \circ A_{1,s(\epsilon)}\right)$ and eq_ϵ the map $B_\epsilon \to A_r$ defined by the equalizer. We also enumerate $E_{ter}(Q) = \{\epsilon_1, \ldots, \epsilon_m\}$.

$$\begin{tikzcd}
& B_{\epsilon_2} \arrow[dr, "eq_{\epsilon_2}"] & & A_2 \arrow[dd] \arrow[dd, dashed, bend left, "\epsilon_1"] \\
L \arrow[ur] \arrow[dr] \arrow[rr, "\phi"] & & A_1 = A_r \arrow[ur] & \\
& B_{\epsilon_1} \arrow[ur, "eq_{\epsilon_1}"] & & A_3 \arrow[ul, dashed, "\epsilon_2"]
\end{tikzcd}$$

Let's prove that L is a cone of D. By definition of the maps, it is already a cone of D'. Let $\epsilon \in E_{ter}(Q)$, we only need to check that, $\phi \circ A_{1,t(\epsilon)} = \phi \circ A_{s(\epsilon),t(\epsilon)} \circ A_{1,s(\epsilon)}$. But for some map f we have, $\phi = f \circ \mathrm{eq}_\epsilon$. By definition of the equalizer $\mathrm{eq}_\epsilon \circ A_{1,t(\epsilon)} = \mathrm{eq}_\epsilon \circ A_{s(\epsilon),t(\epsilon)} \circ A_{1,s(\epsilon)}$, which concludes that L is a cone of D.

Let $(C, (\psi_v)_v)$ be another cone of D. By universal property of equalizers, for all $1 \leqslant i \leqslant m$, there is a unique $C \overset{\alpha_i}{\to} B_{\epsilon_i}$ such that $\mathrm{eq}_{\epsilon_i} \circ \alpha_i = \psi_r$.
Assume that the following diagram (except the dashed arrow) is commutative. Then by property of pullbacks there is a unique $\beta_i : C \to \bigtimes_{A_r}(B_j)_{j\leqslant i}$ which makes the whole diagram commutative.

$$\begin{tikzcd}
C \arrow[rrd, bend left, "\alpha_i"] \arrow[rd, dashed, "\beta_i"] \arrow[rdd, "\beta_{i-1}"'] & & \\
& \bigtimes_{A_r}(B_j)_{j\leqslant i} \arrow[r] \arrow[d] & B_i \arrow[d, "\mathrm{eq}_{\epsilon_i}"] \\
& \bigtimes_{A_r}(B_j)_{j<i} \arrow[r, "\phi_{i-1}"] & A_r
\end{tikzcd}$$

Hence, by induction on $1 \leqslant i \leqslant m$, there is a map $\beta_i : C \to \bigtimes_{A_r}(B_j)_{j\leqslant i}$ such that $\phi_i \circ \beta_i = \psi_r$ thus β_m verifies $A_{1,v} \circ \phi \circ \beta_m = \psi_v$ for all $v \in \{1, \dots, n\}$.
Uniqueness is proved similarly, using at each step of the induction the uniqueness of α_i and then of β_i. □

3 Implementation of the method

3.1 Scope and description

One of our main contributions is the implementation in Python of an algorithm to compute finite (co)limits in the category of finite-dimensional vector spaces. Given a quiver Q and a representation $\mathbf{A}_\bullet$ of Q, the limit is computed following the method described in section 2 and article [SHN21]. As for the colimit, we use the algorithm for limits on the transpose quiver and representation, as suggested by proposition 1.4.

The obtained algorithms work on any field. Classical fields ($\mathbb{R}$, $\mathbb{C}$, $\mathbb{Q}$, $\mathbb{F}_2$) are predefined but custom fields may be added by creating an object "element of the field" and overriding the operators $+, -, -$(unitary), $\times, /, =$. In the case of the fields $\mathbb{R}$ or $\mathbb{C}$, the most time-consuming computations are replaced with faster functions exploiting the properties of $\mathbb{R}$ and $\mathbb{C}$ and the compatibility with the module `numpy`, partly written in C.

Naive implementations using directly formulas from proposition 1.1 and definition 1.7 are also available. They are used as a benchmark in the performance analyses of next subsection and as an empirical verification of the correctness of our algorithms.

In comparison with the description from [SHN21], our algorithm computes separately a limit for each connected component before using the results from subsection 2.5. This idea, although quite simple, brings important gains in time for quivers with few edges.

The theoretical description of section 2 is quite precise. In addition to the implementation of the basic matrix operations, only the computation of the constrained representation spaces in the acyclic reduction step (definition 2.11) needs to be detailed.

These constrained representation spaces, $\mathbf{A}^*_v$, are computed as backwards breadth-first searches starting from the roots of each strongly connected component. Keeping the notations from definition 2.11, when edge $e = (u, v)$ is explored for the i^{th} strongly connected component R_i, the representation space of v has already been replaced by $\bigcap_{j\leqslant i} \Lambda_{v,R_j}$. The representation space of u is then intersected with $\mathbf{A}_e^{-1}\left(\bigcap_{j\leqslant i} \Lambda_{v,R_j}\right)$.

The main basic operations in this algorithm are intersections of linear subspaces and kernels of linear maps. The latter is computed either with a singular value decomposition if the field is $\mathbb{R}$ or $\mathbb{C}$ as in proposition A.5 or by Gaussian elimination of the augmented matrix as in proposition A.6. The intersection of two subspaces spanned by U and V is computed as $\left\{ Uu \mid \begin{pmatrix} u \\ v \end{pmatrix} \in \ker\left(U \quad \mid \quad -V \right) \right\}$. Inverse images are always solved with a Gaussian elimination. In the case of $\mathbb{R}$ or $\mathbb{C}$, entries very close to 0 are regularly flattened to 0 to improve stability and display. The threshold to flatten to 0 is $\epsilon = 10^{-12}$. This parameter is important for the correctness of the algorithm: if $\epsilon > 10^{-10}$, the probability to have a matrix entry in $[-\epsilon, \epsilon]$ is no more negligible, since the number of matrix entries computed in an hour is $\sim 10^{10}$. Similarly, if $\epsilon < 10^{-15}$, numerical errors do occur.

The output of the limit algorithm is:

- the dimension of the limit together with
- the matrices of the maps $\lim \to \mathbf{A}_v$ for $v \in V$.

Since the limit is only defined up to isomorphism, the maps are dependent on a choice of isomorphism from the algorithm. In option, this choice can be made so that the map to the largest representation space has a matrix in echelon form with only unit pivots. Different testing functions are provided including quivers from example 1.2 in the different predefined fields

3.2 Performance analysis

The naive approach implementing the formula from proposition 1.1 uses cubic operations like intersections in the total space which has very high dimension. We will see that the approach described in section 2 will reduce the size of the space in which computations are made, and thus enhance considerably the performances.

To simplify the analysis, we assume that all representation spaces have the same dimension k. If it is not the case, one can use $k = \max_V \dim \mathbf{A}_v$ and obtain upper bounds for the time complexities. We denote $n = |V|$ and $m = |E|$. The basic functions (intersection, kernel, inverse image) which are done through Gaussian pivot or SVD are cubic in the dimensions of the matrix and so are matrix multiplications. Those dimensions are often large as we are dealing with subspaces of the total space. Other operations including the graph searches and matrix constructions will be in practice negligible. As complexities are the same for sections and cosections, we only consider sections. We call naive approach the use of formula 1.1 and we speak of advanced approach for the implementation of the method described in [SHN21] and section 2.

3.2.1 Theoretical time complexities

Rough worst-case complexity bounds for the naive and advanced approaches can be found easily, but we will see later that our algorithms are in practice much faster than the described bounds.

Naive approach The complexity of the naive approach using proposition 1.1 is $\boxed{\mathcal{O}(m(kn)^3)}$. Indeed for $e \in E$, $\ker(\pi_{t(e)} - \mathbf{A}_e \circ \pi_{s(e)})$ is a subspace of the total space of dimension kn. It might be represented by a matrix as big as $kn \times kn$.

Assume that there are $N_{CC} = o(\min(m,n))$ connected components of the same size in the quiver Q. The complexity of the naive approach with separate computations for each connected component becomes $\mathcal{O}\left(m\left(\frac{kn}{N_{CC}}\right)^3\right)$. In reality, this gain is only important when there are few edges, as otherwise NCC is often close to 1.

Strongly connected quiver We consider the complexity of the strongly connected step (subsection 2.2) for a strong quiver of size n,m. There are three phases for this step:

- Finding the ear decomposition which takes $\mathcal{O}(mn)$
- Building the paths in the tree: $|E| - |E_{ter}|$ multiplications i.e. $\mathcal{O}((|E| - |E_{ter}|)k^3)$
- Computing K with $|E_{ter}|$ intersections in $\mathbf{A}_r$: $\mathcal{O}(|E_{ter}|k^3)$

Which gives a total of $\mathcal{O}\left(m(n + k^3)\right)$. However, we expect the constant for the graph search in $\mathcal{O}(mn)$ to be small.

Acyclic reduction The step described in subsection 2.3 is realized through a breadth-first search for each strongly connected component. If we denote N_{SCC} the number of strongly connected components, the cost of computing $\mathbf{A}^*_\bullet$ from the sections of each of the strongly connected components is $\mathcal{O}(N_{SCC} \times mk^3)$.

Arboreal reduction Applying the method of subsection 2.4 to an acyclic quiver of size n,m gives a time complexity $\mathcal{O}\left((m + [V_{max}|)(k|V_{min}|)^3\right)$. Indeed, in 2.17, each edge is associated with a finite number of cubic operations (1 binary equalizer, 1 intersection, 1 matrix multiplication) in the space $\mathbf{A}_r^-$ of dimension $k|V_{min}|$. Moreover, the final computations of proposition 2.18 are $|V_{max}|$ intersections in the same space.

Before looking at the finished algorithm, the first and last steps can already compute limits in the case of respectively strongly connected and acyclic quivers. The time complexity of applying these parts of our algorithm directly to the corresponding special cases is compared to the naive approach in following table.

	Acyclic quiver	**Strongly Connected**	**Naive**
# cubic operations	$\lvert E\rvert + \lvert V_{max}\rvert$	$\lvert E\rvert$	$\lvert E\rvert$
Computation space	$\mathbf{A}_r^- := \prod_{v\in V_{min}} \mathbf{A}_v$	$\mathbf{A}_r$	$\prod_{v\in V} \mathbf{A}_v$
Time complexity	$(\lvert E\rvert + \lvert V_{max}\rvert)k^3\lvert V_{min}\rvert^3$	$\lvert E\rvert k^3 + mn$	$\lvert E\rvert k^3\lvert V\rvert^3$

Advanced approach By summing the complexity of each step, we obtain a total complexity of:

$$\mathcal{O}\left(mn + (m + |V_{max}|)(N_{SCC} + |V_{min}|^3)k^3\right) \tag{2}$$

Let's look in more details at each of the terms of equation (2):

- The term mn, which comes from the strongly connected step is rough upper bound of $\mathcal{O}\left(\sum_{R\in\mathbf{MSC}(Q)} m_R n_R\right)$ where m_R and n_R are the number of edges and vertices of R. For instance if all strongly connected components are of the same size, the complexity of the graph search is $\mathcal{O}\left(\frac{mn}{N_{SCC}}\right)$. Moreover, since the graph search is made of simpler computations, we expect the constant to be small in practice.

- The term $(m + |V_{max}|)N_{SCC}k^3$ is also at most quadratic in the size of the quiver. Furthermore, N_{SCC} is sub-linear in n and m in most examples of families of quivers.

- The term from the arboreal reduction $m|V_{min}|^3k^3$ is the only one which is not quadratic in the size of the quiver. Compared to the naive approach complexity, we have replaced a factor $|V|^3$ by $|V_{min}|^3$. When the are very few edges we have $|V_{min}| \approx |V|$ and, on the contrary, we expect to obtain $|V_{min}| << |V|$ when $m \gg n$. For instance let's take a quiver generated through the Erdős–Rényi directed model (see A.4) with probability $p = \frac{a}{n}$. Then the probability of a vertex u to be minimal is $(1 - p)^{n-1} \approx e^{-a}$ and the expected size of $|V_{min}|$ is ne^{-a}, asymptotically. This means that when we increase linearly m, the number of minimal edges decreases exponentially. In this model, when $m \gg n$, the gains of our algorithm compared to the naive approach will be significant.

In summary, our theoretical upper bound suggests that our algorithm will have a complexity better than a polynomial of degree 4 in the size of the quiver when $m \ll n$ and better than quadratic when $m \gg n$. Unlike the naive approach, the advanced approach seems to be slower when there are only a few edges.

Computing separately each connected component could help overcome this flaw. In the case where $m \ll n$, we have $N_{CC} \approx n$, $|V_{min}| \approx |V|$ and the term $m(|V_{min}k)^3$ is dominant. In this case the gain could be as much as $\frac{1}{n^3}$.

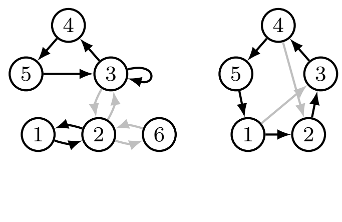
3.2.2 Empirical time complexities

The empirical time complexity is tested on families of random quivers indexed by the number of vertices or of edges. More precisely, on all the following graphs, for each set of parameters (n or m, k, method to generate Q and $\mathbf{A}_\bullet$), $n_{test} = 15$ quivers are generated and the time is averaged on these n_{test} quivers.

Generation of quivers and representations In order to generate random quivers, we will use the directed Erdős–Rényi model (see A.4 as it is one of the simplest and most intuitive. However, to test particular steps of the algorithm, we need graphs with specific properties, namely being strongly connected or acyclic. Since the models are not as intuitive as for general quivers, we provide two different models for each step. All graphs in this essay will be generated with the first models and the second ones will only be used as a verification.

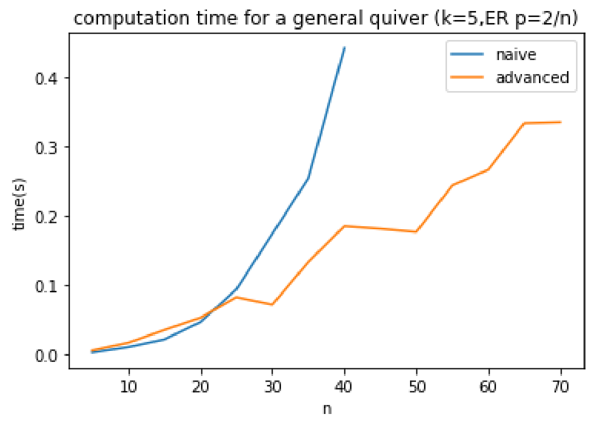
- For acyclic quivers, in the first model, we choose m edges directed in increasing order uniformly at random with repetitions. In the second model, for each of the m edges, we choose uniformly at random the source in $\{1, \ldots, n\}$ and then the target is chosen uniformly in $\{\text{source}, \ldots, n\}$.

- For strongly connected quivers, in the first model we generate a graph with Erdős–Rényi model. Then we select a vertex in each strongly connected component and we add an edge between every couple of selected vertices. In the alternative model, we first create a cycle $1, \ldots, n$, we then add $m - n + 1$ random edges uniformly at random and finally, we shuffle the vertex order.

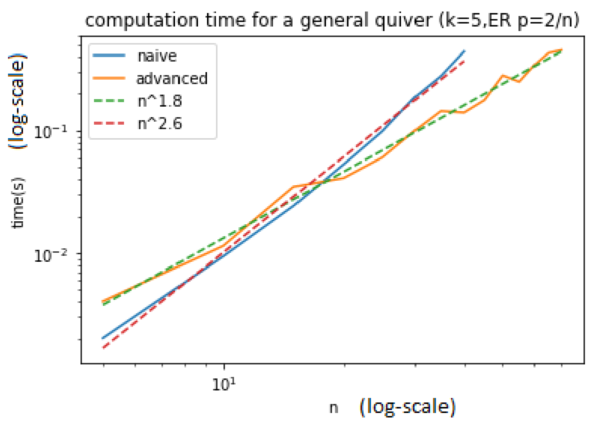


In the performance tests, representation spaces are all real vector spaces of the same dimension k, with most of the time $k = 5$. The maps are generated by perturbing the identity matrix independently for each map. More precisely, a representation map is obtained from the identity by adding 1 at each cell independently with probability $\eta > 0$. η needs to be adjusted carefully. Indeed, if η is too large the limit will nearly always be 0 and if η is too small the limit will most of time be $\prod_{V_{min}} \mathbf{A}_v$. The value $\eta = \frac{1}{km}$ works well empirically.

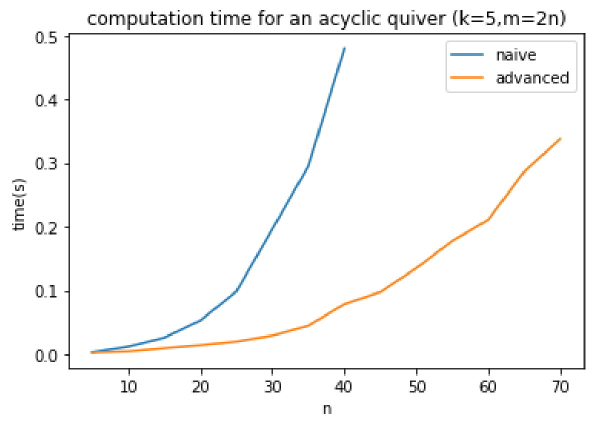
Empirical results To evaluate our algorithm, we perform empirical analyses of computation time for both the naive approach and the advanced approach. We want to know how our algorithm performs for different sizes of quivers (n) and different density (m). The two first experiments show the time analysis when n varies and m is either linear in n for figure 1 or superlinear in n for figure 3. The last one, in figure 4, explores the dependence on m with n fixed.



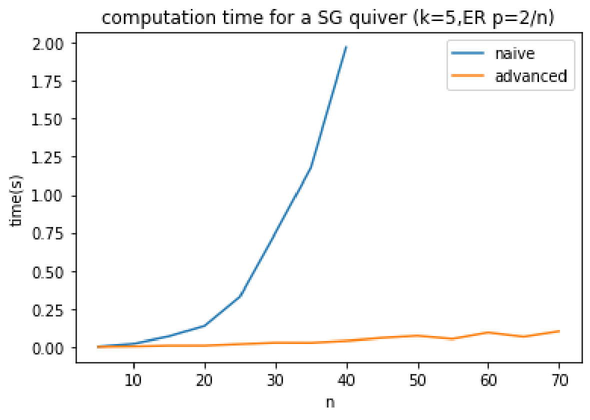
(a) Advanced vs naive approaches



(b) Advanced vs naive in log-scale



(c) Arboreal step



(d) Strongly connected step

Figure 1: Comparison with the naive approach of the general advanced algorithm (1a) and (1b) and of special cases of the advanced algorithm for acyclic (1c) and strongly connected (1d) quivers for a linear number of edges $m = 2n$.

Figure 1 is generated with the parameters described above and a linear number of edges ($m \approx 2n$ or $p = \frac{2}{n}$). Although the model of random quivers is different between (1a), (1c) and (1d), we observe that the arboreal step (subsection 2.4) seems to take as much time as the whole algorithm. This is what we expected for a low number of edges.

The log-log graph of performances (1b) displays an asymptotic performance gain for the advanced algorithm even with only $m = 2n$ edges. Using the least square method in the log-log graph, we obtain an empirical exponent of n in the time complexity of only $n^{1.8}$ and $n^{2.6}$ for, respectively, the advanced and naive approaches.

The empirical complexities are much faster than our theoretical worst-case bounds which is $\mathcal{O}(n^4)$ for both approaches when $m = 2n$. One explanation could be that the intersections to compute are most of time between very simple subspaces: either both spaces to intersect are the same or the bases are block matrices with all blocks being id_k or $0_{k\times k}$. That's why in figure 2, the dimension of the limit goes down nearly always by either 0 or $k = 5$ when a new edge is added to the quiver.

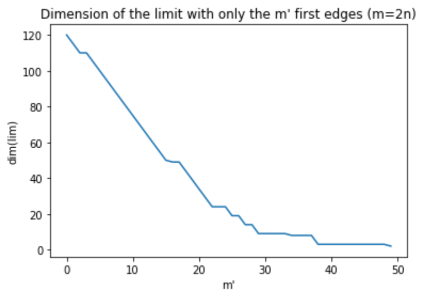
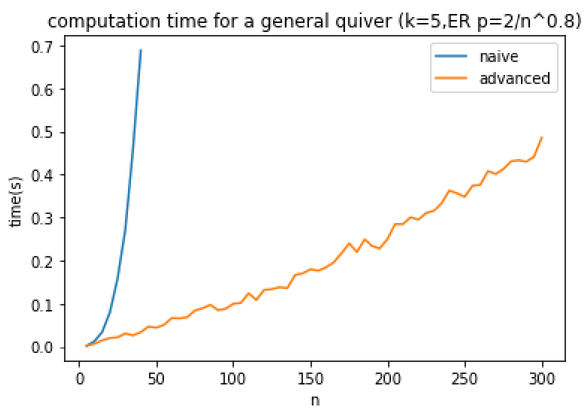
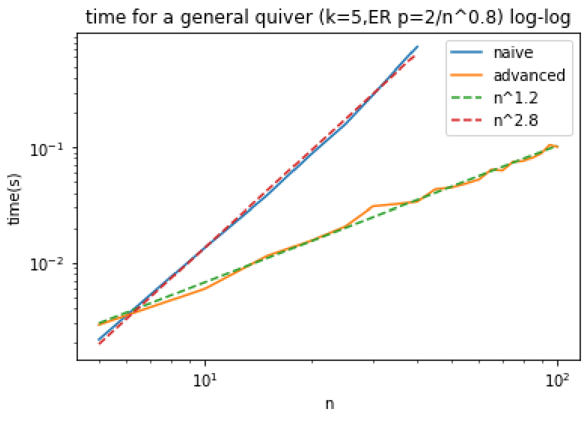


Figure 2: Dimension of the limit for the subquiver with only the $m' \leqslant m$ first edges of a quiver with $m = 2n = 100$ edges

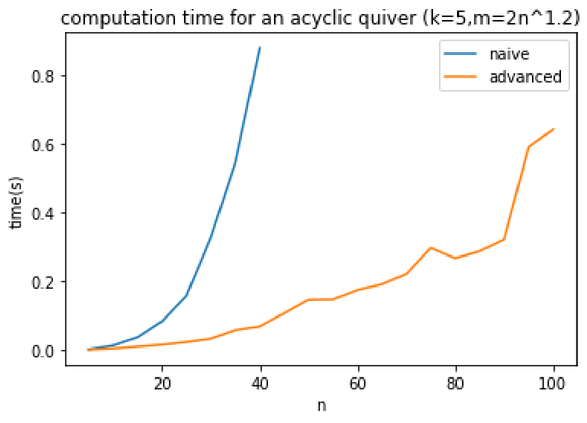
Figure 3 is generated with the same parameters as figure 1 except for the number of edges $m \approx 2n^{1.2}$ (or $p = \frac{2}{n^{0.8}}$). In this situation our algorithm allows to compute in reasonable time quivers nearly 10 times bigger than the naive approach (3a). Comparing with figure 1, the time complexity of the naive approach increased with the number of edges and the exponent in 1b and 3b hints that the increase is linear in m. On the contrary, the advanced algorithm is much faster with denser quivers as the decrease of the dimension of the space of computations, $|V_{min}|k$, allows important speed gains. We also observe that general quivers are easier to compute than random acyclic quivers. This is due to the fact that a quiver with a huge strongly connected component will nearly be an out-tree at the end of the acyclic reduction and thus have much less than $m = n^{1.2}$ edges. Indeed, proposition A.7 shows that our quivers are strongly connected with very high probability when $m \approx n^{1.2}$(or $p = \frac{2}{n^{0.8}}$) but not when $m = 2n$ (or $p = \frac{2}{n}$).



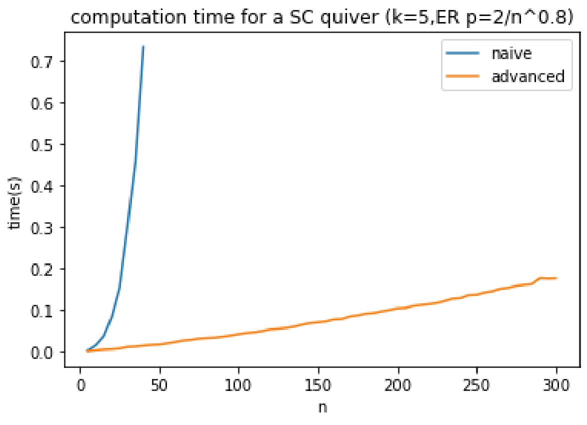
(a) Advanced vs naive approaches



(b) Advanced vs naive in log-scale

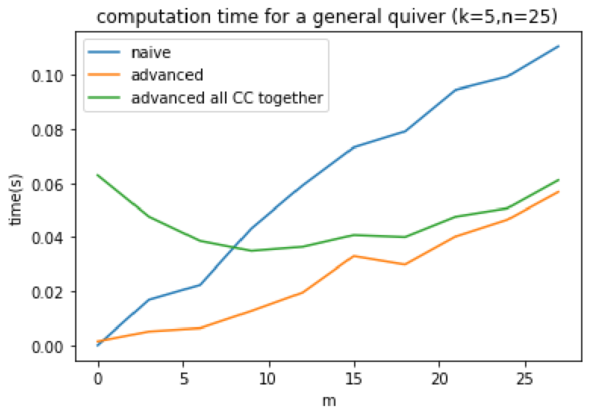


(c) Arboreal step

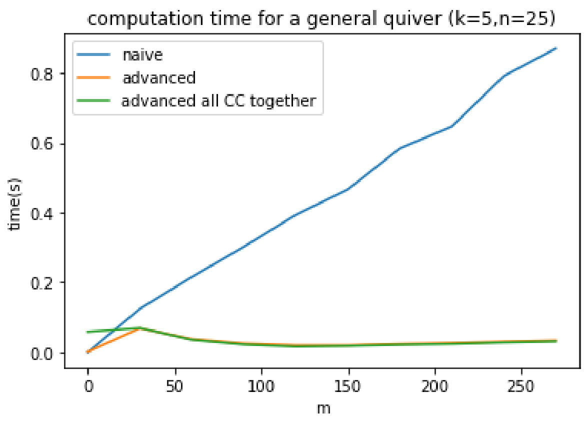


(d) Strongly connected step

Figure 3: Comparison with the naive approach of the general advanced algorithm (1a) and (1b) and of special cases of the advanced algorithm for acyclic (1c) and strongly connected (1d) quivers for a superlinear number of edges $m = 2n^{1.2}$.



(a)



(b)

Figure 4: Comparison between the naive and advanced approaches with or without computing separately each connected components, for $n = 25$ and m varying.

Figure 4 shows the impact of changing the number of edges on the computation time for a quiver with $n = 25$ and $k = 5$. We observe in figure (4a), that the advanced approach without computing separately each connected component (green) is worse than the naive approach for $m \ll n$. However, the advanced approach with separate computations for each component is always better than the naive approach. In figure (4b), one can see that while increasing m makes the naive approach slower almost linearly, it has no significant effect on the advanced approach. More precisely, increasing m makes computations easier when $m \approx n$ as it reduces $|V_{min}|$. On the contrary, when $m \gg n$, $|V_{min}|$ is already close to 1 and increasing m increases the complexity as there are more intersections to compute but the overall time remains small since the computation space is often of dimension only k.

As explained above, the implementation of the most costly operations are optimised for computations in the field $\mathbb{R}$. We see in figure 5 that computations take much longer in $\mathbb{F}_2$ both for the naive approach (green vs blue curbs) and the advanced approach (red vs orange). A log-log analysis shows that the exponent of n in the advanced approach does not increase significantly in $\mathbb{F}_2$ compared to $\mathbb{R}$, which suggest that a large part of the difference could be due to the use of faster C-based `numpy`.

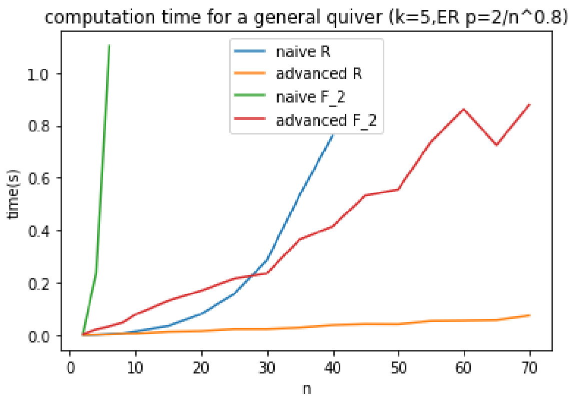


Figure 5: performances in $\mathbb{F}_2$ vs $\mathbb{R}$

Conclusion

Summary The point of view of quivers and representations, helped devise a new method to compute finite (co)limits in the category of finite-dimensional vector spaces. The strategy of this advanced approach, described in [SHN21], is to remove cycles and parallel paths to obtain out(in)-trees, which (co)limits are easy to compute. (Co)limits can also be computed directly, but it involves operations on very large matrices. The advanced approach, which uses smaller matrices, is thus necessary to compute limits of large quivers. My main contribution has been to implement this algorithm. It is in practice much more efficient than the direct computation, especially when the quiver is dense: for real representations, one can compute limits of (not too sparse) quivers with hundreds of vertices.

Outlook This dissertation could be continued in several directions:

- The propositions of section 2 could probably be generalised to any (co)complete category. The general proofs for the simplest propositions are already given in subsection 2.6. One could write and try to prove a generalised version of the entire method.
- The scope of the algorithm could also be extended to other category. We do not expect our algorithm to be easily adaptable to any (co)complete categories. However, it would be interesting to make our algorithm work for modules over a ring and not only for fields.
- One could find a practical example, maybe with cellular sheaves, where our algorithm is useful.

A Annexes

A.1 Quotients of finite-dimensional vector spaces

Cosections involve many quotient computations for finite-dimensional vector spaces. We will recall here some basic properties of these quotients.

Definition A.1 Let $W \subset V \in \mathrm{Vect}_{\mathrm{Fin-Dim}}$, we define $V \big/ W := \{v + W \mid v \in V\}$ and we equip it with the induced finite-dimensional vector space structure.

A useful trick to compute quotients in $\mathrm{Vect}_{\mathrm{Fin-Dim}}$ is that we only need to look at the dimensions:

Proposition A.1 Let $W \subset V \in \mathrm{Vect}_{\mathrm{Fin-Dim}}$, we have $\dim V \big/ W = \dim V - \dim W$.

Proof.

Let W' be a complement of W in V. Then $\left\{ \begin{array}{ll} V \big/ W & \to W' \\ v + W & \mapsto \pi_{W'}(v) \\ w' + W & \leftarrow\!\shortmid w' \end{array} \right.$ is an isomorphism where $\pi_{W'}$ is the projection on W'. □

Corollary A.1 Let $V, V' \in \mathrm{Vect}_{\mathrm{Fin-Dim}}$. We have:

- If $W \subset V$ and $W' \subset V'$ then $V \bigoplus V' \big/ W \bigoplus W' \cong V \big/ W \bigoplus V' \big/ W'$.
- If $W \bigoplus W' \subset V \bigoplus V'$ then

$$V \bigoplus V' \Big/ W \bigoplus W' \cong \left(V \bigoplus V' \Big/ W \right) \Big/ \{w' + W \mid w' \in W'\}.$$

- Let $W \subset V$ and $\phi \in \hom_{\mathrm{Vect}_{\mathrm{Fin-Dim}}}(W, V)$ injective then $V \big/ W \cong V \big/ \phi(W)$

A.2 Elements of category theory

(Co)sections of a quiver with a representation can be seen in the more general context of category theory. This annex contains the definitions and properties from category theory used in this essay especially in sections 1 and 2.6. It is inspired from [Lei16].

Definition A.2 (category). A category $\mathcal{C}$ is the data of

- A collection of objects $\mathrm{ob}(\mathcal{C})$
- For each pair (A, B) of objects a collection of maps $\hom_{\mathcal{C}}(A, B)$ from A to B.
- For each $A, B, C \in \mathrm{ob}\ \mathcal{C}$ a composition map

$$\left\{ \begin{array}{cl} \hom_{\mathcal{C}}(B, C) \times \hom_{\mathcal{C}}(A, B) & \to \hom_{\mathcal{C}}(A, C) \\ (g, f) & \mapsto g \circ f \end{array} \right.$$

- For each $A \in \text{ob } \mathcal{C}$ an element identity $1_A \in \hom_{\mathcal{C}}(A, A)$

with the following properties

- associativity: for each $(f, g, h) \in \hom_{\mathcal{C}}(A, B) \times \hom_{\mathcal{C}}(B, C) \times \hom_{\mathcal{C}}(C, D)$ where $A, B, C, D \in \text{ob } \mathcal{C}$, we have $h \circ (g \circ f) = (h \circ g) \circ f$.
- identity laws: for $A, B \in \text{ob } \mathcal{C}$ and $f \in \hom_{\mathcal{C}}(A, B)$, we have $1_B \circ f = f \circ 1_A = f$.

Definition A.3 A functor F from a category $\mathcal{C}$ to another category $\mathcal{D}$ is

- A function $\left\{ \begin{array}{ll} \text{ob } \mathcal{C} & \to \text{ob } \mathcal{D} \\ A & \mapsto F(A) \end{array} \right.$ and
- For $(A, B) \in \text{ob } \mathcal{C}$ a function $\left\{ \begin{array}{ll} \hom_{\mathcal{C}}(A, B) & \to \hom_{\mathcal{D}}(A, B) \\ f & \mapsto F(f) \end{array} \right.$

with the two properties

- $F(f' \circ f) = F(f') \circ F(f)$ for $A \xrightarrow{f} A' \xrightarrow{f'} A''$ in $\mathcal{C}$
- $F(1_A) = 1_{F(A)}$ for $A \in \text{ob } \mathcal{C}$

Definition A.4 Let $\mathcal{J}$ a category (with a finite number of objects and maps), a (finite) diagram in $\mathcal{C}$ is a functor $D : \mathcal{J} \to \mathcal{C}$. We represent a diagram without writing the identities and the maps obtained by composition. The diagram is said to be commutative if for each $A, B \in \text{ob } \mathcal{J}$, $\text{card} \hom_{\mathcal{C}}(D(A), D(B)) \leqslant 1$; or alternatively, composing along different paths with the same extremities always give the same map.

A quiver $Q = (V, E)$ can be associated to a finite diagram $D : \mathcal{J} \to \mathcal{C}$ by:

- $V := \text{ob } \mathcal{J}$
- $E := \{[A, B] \mid \exists f \in \hom_{\mathcal{J}}(A, B) \text{ s.t. } f \neq 1_A \text{ and } \nexists A \xrightarrow{g} C \xrightarrow{h} B, f = h \circ g\}$

The definition of *finite limits* in a general category $\mathcal{C}$ can be obtained from definition 1.4 by replacing

- "quiver with a representation" by "diagram in $\mathcal{C}$"
- "finite-dimensional vector spaces" by "objects of $\mathcal{C}$"
- "linear maps" by "maps in $\mathcal{C}$".

Important examples includes the ones described in 1.2: products, equalizers, pullbacks. However, in this more general setting, finite limits may not always exist.

Definition A.5 A category where all (finite) limits exist is said to be complete.

Proposition A.2 A category where all products and equalizer exist is complete.

Let $\mathcal{C}$ be a complete category.

Lemma A.1 In the following diagram in $\mathcal{C}$, if both squares are pullbacks then the large rectangle is also a pullback:

$$\begin{tikzcd}
A \arrow[r, "f_1"] \arrow[d, "h_1"] & B \arrow[r, "f_2"] \arrow[d, "h_2"] & C \arrow[d, "h_3"] \\
D \arrow[r, "g_1"] & E \arrow[r, "g_2"] & F
\end{tikzcd}$$

Proof.

- The two square being commutative, the large rectangle is commutative:

$$\begin{aligned} h_3 \circ f_2 \circ f_1 &= g_2 \circ h_2 \circ f_1 && \text{since the right square is a pullback} \\ &= g_2 \circ g_1 \circ h_1 && \text{since the left square is a pullback} \end{aligned}$$

  Hence, $(A, f_2 \circ f_1, h_1)$ is a cone of the diagram $D \to F \leftarrow C$:

- Let $(A', A', \overset{f'}{\to} C, \overset{h'}{\to} D)$ be another cone of $D \to F \leftarrow C$. $(A', A', \overset{f'}{\to} C, \xrightarrow{g_1 \circ h'} E)$ is a cone of $E \to F \leftarrow C$. Since the right square is a pullback, there is a unique map $A' \overset{\alpha}{\to} B$ such that the following diagram commutes:

$$\begin{tikzcd}
A' \arrow[rrrd, bend left, "f'"] \arrow[rrd, dashed, "\alpha"] \arrow[rdd, bend right, "h'"'] & & & \\
& A \arrow[r, "f_1"] \arrow[d, "h_1"] & B \arrow[r, "f_2"] \arrow[d, "h_2"] & C \arrow[d, "h_3"] \\
& D \arrow[r, "g_1"] & E \arrow[r, "g_2"] & F
\end{tikzcd}$$

  But now (A', α, h') is a cone of $D \to E \leftarrow B$ and there is a unique map $A' \overset{\beta}{\to} A$ such that $\alpha = f_1 \circ \beta$ and $h' = h_1 \circ \beta$.
  Hence $f' = f_2 \circ f_1 \circ \beta$ and $h' = h_1 \circ \beta$. Moreover, if β' verifies $f' = f_2 \circ f_1 \circ \beta'$ and $h' = h_1 \circ \beta'$, by uniqueness of α, $f_1 \circ \beta' = \alpha$ and by uniqueness of β, $\beta' = \beta$.

□

Proposition A.3 (Associativity of pullbacks). Let $A, B, C, D, E \in \text{ob}(\mathcal{C})$, there is an isomorphism $(A \times_B C) \times_D E \cong A \times_B (C \times_D E)$.

Proof. Let's apply lemma A.1 to the horizontal and vertical rectangles of the following diagram made of pullback squares:

$$\begin{tikzcd}
P \arrow[r] \arrow[d] & C \times_D E \arrow[r] \arrow[d] & E \arrow[d] \\
A \times_B C \arrow[r] \arrow[d] & C \arrow[r] \arrow[d] & D \\
A \arrow[r] & B &
\end{tikzcd}$$

We obtain that $P \cong A \times_B (C \times_D E)$ (vertical rectangle) and $P \cong (A \times_B C) \times_D E$ (horizontal rectangle), which means $A \times_B (C \times_D E) \cong (A \times_B C) \times_D E$. □

It follows that we can write $\bigtimes_B (A_i)_{i \in I}$ for a finite set I and $B, A_i \in \text{ob}(\mathcal{C})$ without worrying about the order.

The dual notions - colimits, cocomplete, coproduct, coequalizer, pushforward - can be defined in a similar fashion.

A.3 Elements of numerical linear algebra

In this annex inspired from [TB97], the methods from numerical linear algebra used in our algorithm are explained especially SVD and Gaussian elimination.

Proposition A.4 (SVD). Let $m, n \geqslant 0$ and $A \in \mathbb{C}^{m\times n}$, then there is $U \in \mathbb{C}^{m\times m}$ and $V \in \mathbb{C}^{n\times n}$ two unitary matrices and $\Sigma \in \mathbb{R}^{m\times n}$ a diagonal matrix with entries $\sigma_1 \geqslant \sigma_2 \geqslant \cdots \geqslant \sigma_{\min(m,n)} \geqslant 0$ such that

$$A = U\Sigma V^*.$$

Moreover, if M is a real matrix, U and V are also real.

Proof. Let's prove it by induction on m. If $m = 1$, $A = (||A||_2)A^*$. If $m > 1$ let $\sigma_1 = ||A||_2$, by compacity, there is $u_1 \in \mathbb{C}^m$ and $v_1 \in \mathbb{C}^n$ of of norm 1 such that $Av_1 = \sigma_1 u_1$. We complete u and v into orthogonal basis U_1 and V_1 of respectively C^m and $\mathbb{C}^n$. Then, $U_1^* A V_1 = \begin{pmatrix} \sigma_1 & w^* \\ 0 & A' \end{pmatrix}$ where $w \in \mathbb{C}^{m-1}$ and $B \in \mathbb{C}^{(m-1)\times(n-1)}$.

$$\sigma_1 = ||A||_2 = ||U_1 A V_1^*||_2 \geqslant \left\| \begin{pmatrix} \sigma_1 & w^* \\ 0 & A' \end{pmatrix} \begin{pmatrix} \sigma_1 \\ w \end{pmatrix} \right\|_2 / \left\| \begin{pmatrix} \sigma_1 \\ w \end{pmatrix} \right\|_2 \geqslant \sqrt{\sigma_1^2 + [[w||_2^2}$$

So $w = 0$. By applying the induction hypothesis to A', $A' = U'\Sigma' V'^*$ with U' and V' unitary and Σ' diagonal with real nonpositive entries in decreasing order. Finally,

$$A = \underbrace{U_1 \begin{pmatrix} 1 & 0 \\ 0 & U' \end{pmatrix}}_{U} \underbrace{\begin{pmatrix} \sigma_1 & 0 \\ 0 & \Sigma' \end{pmatrix}}_{\Sigma} \underbrace{\begin{pmatrix} 1 & 0 \\ 0 & V^{*\prime} \end{pmatrix} V_1^*}_{V^*}.$$

The proof still holds when A is real and $\mathbb{C}$ is replaced everywhere by $\mathbb{R}$. □

Proposition A.5 Let $A \in C^{m\times n}$ with its SVD decomposition $A = U\Sigma V$. Then if r is the rank of A (and the last nonzero column of Σ), the columns $r+1, \ldots, n$ of V constitutes a basis of the kernel of A.

Proof. The last $n - r$ columns of $AV = U\Sigma$ are zeros. □

This methods only works for $\mathbb{R}$ or $\mathbb{C}$ as it relies on square roots/absolute value of reals. Its complexity is $\mathcal{O}(n^2(m+n))$. The nullspace obtained has some stability properties as very small singular values can be considered as 0.

Proposition A.6 (Gaussian elimination and kernel). Let $A \in C^{m\times n}$. We put the augmented matrix $\begin{pmatrix} A \\ I_m \end{pmatrix}$ into column echelon form $\begin{pmatrix} AP \\ P \end{pmatrix}$ where P is invertible.

The nullspace of A is spanned by the columns of P corresponding to zero-columns of AP.

This method works with any field its complexity is $\mathcal{O}(n^2(m+n))$. As explained in [TB97], when partial pivoting is used this method is stable in practice.

A.4 Directed Erdős–Rényi model

Directed random graphs are used to test our algorithm. This annex contains the definition of the model alongside an interesting proposition on strong connectedness proved in article [GP08].

Definition A.6 (Erdős–Rényi for digraphs). In the directed Erdős–Rényi model with parameter n, the vertices are $1, \ldots, n$ and for each $i \neq j$ the edge (i, j) is chosen independently with probability p.

Proposition A.7 For any $\epsilon > 0$, asymptotically a directed Erdős–Rényi graph is

- strongly connected if $p > (1+\epsilon)\frac{\ln n}{n}$
- not strongly connected if $p < (1-\epsilon)\frac{\ln n}{n}$

More precisely, if (G_n) is a family of directed Erdős–Rényi graphs with parameters $(n, p(n))_{n\in\mathbb{N}}$, then $\lim_\infty \mathbb{P}(G_n \text{ is strongly connected})$ is 1 if $p(n) > (1+\epsilon)\frac{\ln n}{n}$ and 0 if $p(n) < (1-\epsilon)\frac{\ln n}{n}$ for n large enough.

B Python code

The Python code corresponding to the algorithm to compute limits is provided below. It contains neither the dual algorithm for the cosections space nor the testing functions used to generate the graphs of the empirical performance analysis. The code is organized in 5 files: a main, 3 files for the 3 steps of the algorithm and a file for low-level functions.

`main_simple.py`

```python
#linear algebra
import numpy as np
from aux_fun_simple import proj, intersection,null_space
#quiver and matrix in fields
from aux_fun_simple import Field,eye_mat,Quiver
#display
from aux_fun_simple import print_limit
#steps of advanced algo
from arboreal_simple import arboreal_out
from acycli_red_simple import acyclic_red
#graph
import networkx as nx


epsilon=1e-12

## NAIVE APPROACH
def sections_naive(Q):
    field=Q.field
    #compute partial dimensions in \sum_v Av
    partial_sum_Av=[0]
    for i in range(Q.n):
        partial_sum_Av.append(partial_sum_Av[-1]+Q.Av[i])
    #Initialize Gamma as the total space
    Gamma= eye_mat(partial_sum_Av[-1],field)

    for i_e,e in enumerate(Q.E):
        #build projections
        pi_se=proj(partial_sum_Av[e[0]],Q.Av[e[0]],partial_sum_Av[-1],field)
        pi_te=proj(partial_sum_Av[e[1]],Q.Av[e[1]],partial_sum_Av[-1],field)
        #update Gamma
        Gamma = intersection(Gamma,null_space(pi_te-np.matmul(Q.Ae[i_e],pi_se),field),field=field)
    return len(Gamma[0])

## ADVANCED APPROACH
def compute_sections(Q,CC_separated=True):
    field=Q.field
    #if all connected components are computed together
    if not(CC_separated):
        Q_star,Av_star=acyclic_red(Q)
        return arboreal_out(Q_star,True)
    #compute the weakly connected components
    G=nx.DiGraph()
    G.add_nodes_from(range(Q.n))
    G.add_edges_from(Q.E)
    weak_compos,sub_nodes,sub_edges=[],[],[]
    ind_inv=[[-1,-1] for _ in range(Q.n)] #reordering
    for i_compo,compo in enumerate(nx.weakly_connected_components(G)):
        #build subgraphs
        sub_nodes.append(np.array(list(compo)))
        sub_edges.append([i_e for i_e,e in enumerate(Q.E) if (e[0] in compo and e[1] in compo)])
        #keep track of labels
        for i_v,v in enumerate(sub_nodes[-1]):
            ind_inv[v]=[i_compo,i_v]
        #relabel
        renamed_edges=[[ind_inv[Q.E[i][0]][1],ind_inv[Q.E[i][1]][1]] for i in sub_edges[-1]]
        weak_compos.append(Quiver(len(sub_nodes[-1]),renamed_edges,
                                  [Q.Av[x] for x in sub_nodes[-1]],[Q.Ae[x] for x  in sub_edges[-1]]
    #section computation by weakly connected compo
    lim,maps=[],[]
    for Q_c in weak_compos:
        Q_star,Av_star=acyclic_red(Q_c)#acyclic reduction
        lim_c,maps_c=arboreal_out(Q_star,True) #arboreal reduction
        #reconstruction of the maps
        for i in range(len(maps_c)):
            if maps_c[i].shape[0]*maps_c[i].shape[1]==0:
                maps_c[i]=np.zeros((Q_c.Av[i],lim_c))
            else:
                maps_c[i]=np.dot(Av_star[i], maps_c[i])
        lim.append(lim_c)
        maps.append(maps_c)
    # total limit and maps from the limit and maps of each CC
    lim_final=sum(lim)
    maps_final=[]
    for v in range(Q.n):
        i_compo=ind_inv[v][0]
        blocks=[np.zeros((Q.Av[v],sum(lim[:i_compo]))), maps[i_compo][ind_inv[v][1]]]
        maps_final.append(np.concatenate(blocks,axis=1))
    return lim_final,maps_final


## TEST
def test_classics():
    print("=====================================================")
    print("pullback")
    Q=Quiver(3,[[0,2],[1,2]],[3,2,2],[np.array([[1,0,1],[1,1,0]]),np.eye(2)],Field('R'))
    print(Q)
    print_limit(compute_sections(Q))

    print("=====================================================")
    print("equalizer")
    Q=Quiver(2,[[0,1],[0,1]],[3,3],[np.array([[1,0,1],[0,1,0],[0,1,1]]),np.array([[1,1,0],[0,1,0],[0
    print(Q)
    print_limit(compute_sections(Q))

    print("=====================================================")
    print("different fields")
    for field in [Field("Q"),Field("F_2")]:
        mat_11_temp=eye_mat(2,field)
        mat_11_temp[0][0]= field.one+field.one+field.one
        Q=Quiver(2,[[0,1],[1,1]],[2,2],[eye_mat(2,field),mat_11_temp],field)
        print(Q)
        print_limit(compute_sections(Q,field),field)

test_classics()
```

arboreal_simple.py

```
#linear algebra
from aux_fun_simple import intersection,proj,null_space,flatten_zero
#quiver and matrix in fields
from aux_fun_simple import E_in_out,shift_vertices,zeros_mat,eye_mat
import numpy as np

## Reordering vertices so that edges are increasing
def swap_int(x,u,v):
    if x==u:
        return v
    if x==v:
        return u
    return x


def swap_vertices(Q,u,v):
    Q.Av[u],Q.Av[v]=Q.Av[v],Q.Av[u]
    Q.E=[[swap_int(e[0],u,v),swap_int(e[1],u,v)] for e in Q.E]


def order_vertices(Q):
    order=list(range(Q.n))
    i_e=0
    while i_e<len(Q.E):
        cur_edge=Q.E[i_e]
        if cur_edge[0]>cur_edge[1]:
            swap_vertices(Q,cur_edge[0],cur_edge[1])
            order=[swap_int(x, cur_edge[0], cur_edge[1]) for x in order]
            #print("swap",cur_edge[0],cur_edge[1],order)
            i_e=-1
        i_e+=1
    return order

## sections of an acyclic quiver
def arboreal_out(Q,maps):
    #solve separately the trivial quiver for speed when m<<n
    if Q.n==1 and len(Q.E)==0:
        if maps:
            return Q.Av[0], [np.eye(Q.Av[0])]
        return Q.Av[0]
    #reorder V to make edges increasing
    order=order_vertices(Q)
    #add a root
    Q=shift_vertices(Q)
    order=[x+1 for x in order]

    field=Q.field
    Phi=[np.array([]) for _ in range(Q.n)]
    phi=[np.array([]) for _ in range(Q.n)]
    E_in,E_out=E_in_out(Q)
    #computing minimal and maximal vertices
    V_min,V_max=[],[]
    for v in range(1,Q.n):
        if E_out[v]==[]:
            V_max.append(v)
        if E_in[v]==[]:
            V_min.append(v)
    Av_min=[Q.Av[v] for v in V_min]
    sum_Av_min=sum(Av_min)
    #special case: root of dim 0
    if sum_Av_min==0:
        if maps:
            return 0, [np.array([]).reshape(Q.Av[i],0) for i in range(1,Q.n)]
        else:
            return 0
    # representation of the root
    Q.Av[0]=sum_Av_min
    Phi[0]= eye_mat(sum_Av_min,field)
    phi[0]= eye_mat(sum_Av_min,field)
    #maps root-> minimal vertices
    sum_Av_min_p=0
    for i in range(len(V_min)):
        Phi[V_min[i]] = eye_mat(sum_Av_min,field)
        phi[V_min[i]] = proj(sum_Av_min_p,Av_min[i],sum_Av_min,field)
        sum_Av_min_p+=Av_min[i]
    #computation if sections by going down the graph
    for v in range(1,Q.n):
        for e in E_out[v]:
            u=Q.E[e][1]#current vertex
            #if u not seen
            if len(Phi[u])==0:
                Phi[u]=Phi[v]
                #dimension 0
                if Q.Ae[e].shape[1]==0:
                    phi[u]=zeros_mat(Q.Av[u],phi[v].shape[1],field)
                else:
                    phi[u]= np.dot(Q.Ae[e],phi[v])
            else:
                if Q.Ae[e].shape[1]==0:#dim 0
                    equali=phi[u]#difference of functions in equalizer
                else:
                    equali=phi[u]-np.dot(Q.Ae[e],phi[v])
                equali= flatten_zero(equali,field)
                if Q.Av[u]!=0:
                    Phi[u]=intersection(null_space(equali,field),intersection(Phi[v],Phi[u],field),f
    #compute total flow space
    result_space = Phi[0]
    for v in V_max:
        result_space=intersection(result_space,Phi[v],field)
    if maps:
        #return dim(Av*) and isomorphism field^dim(Av*)-> subspace of Av
        result_maps=[]
        for v in range(0,Q.n-1):
            result_maps.append(np.dot(phi[order[v]],result_space))
        return  result_space.shape[1],result_maps
    return len(result_space[0])
```

acycli_red_simple.py

```
#linear algebra
from aux_fun_simple import intersection,row_echelon,col_echelon,solve_triangular,null_space
#Quiver and matrices in fields
from aux_fun_simple import Quiver,E_in_out,zeros_mat,eye_mat,is_all_zero_mat
from strongly_connected_simple import SG_to_tree
import numpy as np
import networkx as nx
import copy
epsilon=1e-12

# compute the inverse image of M restricted to Im(M)\cap K
def inverse_image(M,K,field):
    #column echelon form
    Img_M=col_echelon(M,field)
    #eliminate zero columns
    if field.descr in ['R','C']:
        non_zero_cols=np.where(np.max(np.abs(Img_M),axis=0)>epsilon)[0]
    else:
        non_zero_cols=[i  for i,M_col_i in enumerate(list(Img_M.transpose())) if not(is_all_zero_mat
    Img_M=Img_M[:,non_zero_cols]
    #basis of Im(M)\cap K
    Img_inter=intersection(Img_M,K,field)
    #solve Mx=y for y in Im(M)\cap K
    x=inverse_image_vect(M,Img_inter,field)
    ker = null_space(M,field)#add a basis of ker(M)
    return np.concatenate((x, ker),axis=1)

#solve Mx=y
def inverse_image_vect(M,y,field):
    #empty matrix
    if M.shape[1]*M.shape[0]==0:
        return np.array([]).reshape(M.shape[1],0)
    #column echelon form of the augmented matrix
    Augmented_mat,pivots=row_echelon(np.concatenate((M,y),axis=1),field)
    M_ech=Augmented_mat[:,:len(M[0])]
    #no solution
    if len(pivots)>0 and pivots[-1]>= np.shape(M)[1]:
        return np.array([]).reshape(np.shape(M)[0],0)
    #tranform into square invertible triangular matrix and solve
    non_zero_rows = np.array([i for i in range(len(M_ech)) if not(is_all_zero_mat(M_ech[i], field))]
    x_part= solve_triangular(M_ech[non_zero_rows][:,np.array(pivots)],Augmented_mat[non_zero_rows][:
    #reintegrate 0 rows and non pivots
    x=zeros_mat(len(M[0]),len(x_part[0]),field)
    for i_p,p in enumerate(pivots):
        x[p]=x_part[i_p]
    return x



def acyclic_red(Q):
    field=Q.field
    #find strongly connected components
    G=nx.DiGraph()
    G.add_nodes_from(range(Q.n))
    G.add_edges_from(Q.E)
    SG_compo,sub_nodes,sub_edges=[],[],[]
    ind_inv=[[-1,-1] for _ in range(Q.n)]#keep track of the decomposition
    for i_compo,compo in enumerate(nx.strongly_connected_components(G)):
        sub_nodes.append(np.array(list(compo)))
        sub_edges.append([i_e for i_e,e in enumerate(Q.E) if (e[0] in compo and e[1] in compo)])

        for i_v,v in enumerate(sub_nodes[-1]):
            ind_inv[v]=[i_compo,i_v]
        renamed_edges=[[ind_inv[Q.E[i][0]][1],ind_inv[Q.E[i][1]][1]] for i in sub_edges[-1]]
        SG_compo.append(Quiver(len(sub_nodes[-1]),renamed_edges,
                               [Q.Av[x] for x in sub_nodes[-1]],[Q.Ae[x] for x  in sub_edges[-1]]
    Q_star=Q
    #apply SG_to_tree to each SC component
    roots,E_star,Av_star,AE_star=[],[],[eye_mat(k,field) for k in Q.Av],[]
    for i_R,R in enumerate(SG_compo):
        K,T,r=SG_to_tree(R,True)
        roots.append(sub_nodes[i_R][r])
        E_star.extend([[sub_nodes[i_R][e[0]],sub_nodes[i_R][e[1]]] for e in T.E])
        Av_star[roots[-1]]=K
        AE_star.extend(T.Ae)
    #add edges not in a stronggly connected component
    for i_e,e in enumerate(Q.E):
        if ind_inv[e[0]][0]!=ind_inv[e[1]][0]:
            E_star.append(e)
            AE_star.append(Q.Ae[i_e])
    #transform Av,Ae into a valid representation Av*,Ae*
    Q_star=Quiver(Q.n,E_star,Q.Av,AE_star,field)
    E_in,E_out=E_in_out(Q_star)
    for i_R,R in enumerate(SG_compo):
        stack=[roots[i_R]]
        while len(stack)!=0:
            v=stack.pop()
            for i_e in E_in[v]:
                Av_star[Q_star.E[i_e][0]]=intersection(Av_star[Q_star.E[i_e][0]],inverse_image(copy.
                if not(Q_star.E[i_e][0] in stack):
                    stack.append(Q_star.E[i_e][0])
    # transform from subspaces of Av to field^d
    Av_star_dim=[len(A[0]) for A in Av_star]
    Ae_star=[inverse_image_vect(Av_star[Q_star.E[i_y][1]], np.dot(y,Av_star[Q_star.E[i_y][0]]),field
    for i in range(len(Ae_star)):
        if Ae_star[i].shape[0]*Ae_star[i].shape[1]==0:
            Ae_star[i]=np.zeros((Av_star_dim[Q_star.E[i][1]],Av_star_dim[Q_star.E[i][0]]))
    Q_star.Av=Av_star_dim
    Q_star.Ae=Ae_star
    return Q_star,Av_star
```

strongly_connected_simple.py

```
import numpy as np
#linear algebra
from aux_fun_simple import intersection,null_space
#Quiver and matrices in fields
from aux_fun_simple import  E_in_out,Quiver,eye_mat



#build ear decomposition of a SG quiver
def find_ear_decompo(Q):
    E_in,E_out=E_in_out(Q)
    v_ear=-1*np.ones(Q.n,dtype=int)#all vertices not seen
    e_ear=-1*np.ones(len(Q.E),dtype=int)# all edges not seen

    #find a cycle
    start=0
    v_ear[start]=0
    e_ear[E_out[start][0]]=0
    v=  Q.E[ E_out[start][0]][1]
    l=[start]#visited vertices
    while(v_ear[v]!=0):
        l.append(v)
        #mark as seen
        v_ear[v]=0
        e_ear[E_out[v][0]]=0
        #go to next vertex
        v=  Q.E[ E_out[v][0]][1]

    r=v#first point of the cycle
    #mark as unseen vertices not in the cycle
    for u in l[:l.index(r)]:
        v_ear[u]=-1
        e_ear[E_out[u][0]]=-1
    #build ear decomposition
    ear_num=0#last built ear
    while(len(np.where(v_ear==-1)[0])!=0):
        #edge outgoing from decompo
        e0=None
        for i_e,e in enumerate(Q.E):
            if v_ear[e[0]]>=0 and v_ear[e[1]]==-1:
               e0=i_e
               break
        w=Q.E[e0][1]
        #shortest pasth w-> ear_num
        distances=(2*Q.n+1)*np.ones(Q.n)
        previous_edge=[[] for _ in range(Q.n)]#shortest paths from w to x
        d=0#distance to w
        distances[w]=d
        previous_edge[w]=e0
        e_ear[e0]=ear_num+1
        arrived=False
        while(not arrived):
            for w_cur in  np.where(distances==d)[0]:
                for e_cur in E_out[w_cur]:
                    #if new vertex in the ear decompo choose this path
                    if not(arrived) and v_ear[Q.E[e_cur][1]]!=-1:
                        arrived=True
                        #retrieve the path inductively
                        e_backwards=e_cur
                        while (v_ear[Q.E[e_backwards][0]]==-1):
                            v_ear[Q.E[e_backwards][0]]=ear_num+1
                            e_ear[e_backwards]=ear_num+1
                            e_backwards=previous_edge[Q.E[e_backwards][0]]
                        ear_num+=1
                        break
                    #continue search
                    if distances[Q.E[e_cur][1]]>d+1:
                        distances[Q.E[e_cur][1]]=d+1
                        previous_edge[Q.E[e_cur][1]]=e_cur

                if arrived:
                    break
            if arrived:
                break
            d+=1
    for e in np.where(e_ear==-1)[0]:
        e_ear[e]=ear_num+1
        ear_num+=1
    return r,v_ear,e_ear


#strongly connected quiver to out-tree
def SG_to_tree(Q,only_K):
    field=Q.field#retrieve field
    #trivial quiver
    if len(Q.E)==0:
        if only_K:
            return eye_mat(Q.Av[0],field),Q,0
        return Q
    #choose an ear decomposition
    r,v_ear,e_ear=find_ear_decompo(Q)
    #find terminal edges
    ter_edges=[]
    non_ter_edges=[]
    for i_e,e in enumerate(Q.E):
        if e_ear[i_e]>v_ear[e[1]] or e[1]==r:
            ter_edges.append(i_e)
        else:
            non_ter_edges.append(i_e)

    #build out-tree
    T=Quiver(Q.n,[Q.E[i_e] for i_e in non_ter_edges],Q.Av,[Q.Ae[i_e] for i_e in non_ter_edges],field
    E_in_T,E_out_T=E_in_out(T)
    #build paths from root
    phi=[[] for _ in range (Q.n)]
    phi[r]=eye_mat(Q.Av[r],field)
    stack=[r]
    while not len(stack)==0:
        v=stack.pop()
        for i_e in E_out_T[v]:
            phi[T.E[i_e][1]]=np.dot(T.Ae[i_e],phi[v])
            stack.append(T.E[i_e][1])
    #compute K
    K=eye_mat(Q.Av[r],field)
    for epsilon in ter_edges:
        K=intersection(K,null_space(phi[Q.E[epsilon][1]]-np.dot(Q.Ae[epsilon],phi[Q.E[epsilon][0]]),
    if only_K:
        return K,T,r
    #return as a quiver
    T.Av[r]=len(K[0])
    for i_e in E_out_T[r]:
        T.Ae[i_e]=np.dot(T.Ae[i_e],K)
    return T
```

aux_fun_simple.py

```
import numpy as np
import scipy.linalg
import copy
from fractions import Fraction
Q_pb=[]


#maximum computation error
epsilon=1e-12

## COMPUTATIONS IN ALL FIELDS

#field F_2
class F_2():
    #definition
    is_one=False
    def __init__(self, is_one):
        self.is_one=is_one
    #override operators
    def __add__(self,b):
        return F_2(self.is_one^b.is_one)
    def __sub__(self,b):
        return self+b
    def __mul__(self,b):
        return F_2(self.is_one&b.is_one)
    def __truediv__(self,b):
        if not(b.is_one):
            raise ValueError("Divide by 0")
        else:
            return self
    def __neg__(self):
        return self
    def __eq__(self,b):
        return isinstance(b, F_2) and self.is_one==b.is_one
    #display
    def __str__(self):
        return "cl("+str(int(self.is_one))+")"
    def __repr__(self):
        return "cl("+str(int(self.is_one))+")"

#general field definition
class Field:
    descr='O'
    zero=0.
    one=1.
    def __init__(self,*args):
        # usual fields R, C, F_2, Q
        if len(args)==1:
            if args[0] in ['R','C']:
                self.descr=args[0]
            elif args[0]=='F_2':
                self.descr=args[0]
                self.zero=F_2(0)
                self.one=F_2(1)
            elif args[0]=='Q':
                self.descr=args[0]
                self.zero=Fraction(0)
                self.one=Fraction(1)
            else:
                raise ValueError("No predefined field "+args[0])
        #user-defined fields
        else:
            self.zero=args[0]
            self.one=args[1]
            if len(args)==3:
                self.descr=args[2]

#Replacing numpy operations if the field is not R or C
#np.zeros
def zeros_mat(n,m,field):
    if field.descr=='R':
        return np.zeros((n,m))
    if field.descr=='R':
        return np.zeros((n,m),dtype=complex)
    return np.array([[field.zero for _ in range(m)]for _ in range(n)]).reshape((n,m))

#np.eye
def eye_mat(n,field):
    if field.descr =='R':
        return np.eye(n)
    if field.descr=='C':
        return np.eye(n,dtype=complex)
    I=zeros_mat(n,n,field)
    for i in range(n):
            I[i][i]=field.one
    return I

def is_all_zero_mat(M,field):
    if field.descr in ['R','C']:
        return np.max(np.abs(M))<epsilon
    return all([m==field.zero for m in M.flatten()])

def is_all_zero_elem(x,field):
    return is_all_zero_mat(np.array([x]),field)
#Remove computation errors
def flatten_zero(U,field):
    if field.descr in ['R','C']:
        V=U.flatten()
        V[np.where(np.abs(V)<epsilon)[0]]=0.
        return V.reshape(U.shape)
    return U

## QUIVER
#definition
class Quiver:
  field='R'
  def __init__(self, n,edges,rep_spaces,rep_maps,field):
    self.n = n #|V|
    self.E = edges
    self.Av=np.array(rep_spaces,dtype=int) #dimension of A_v
    self.Ae=rep_maps
    self.field=field
  #display
  def __str__(self):
      if self.field.descr in ['R','C']:
          Ae_rounded=[np.round(ae,3) for ae in self.Ae]
          return "n="+str(self.n)+" E="+str(self.E)+" Av="+str(self.Av)+"\n Ae="+str(Ae_rounded)
      return "n="+str(self.n)+" E="+str(self.E)+" Av="+str(self.Av)+"\n Ae="+str([self.Ae])
  def __repr__(self):
      if self.field.descr in ['R','C']:
          return "n="+str(self.n)+" E="+str(self.E)+" Av="+str(self.Av)+"\n Ae="+str([np.round(ae,3)
      return "n="+str(self.n)+" E="+str(self.E)+" Av="+str(self.Av)+"\n Ae="+str(self.Ae)

#add new vertices with order 0,1,...
def shift_vertices(Q,shi=1):
    E=[]
    for e in Q.E:
        E.append([e[0]+1,e[1]+1])
    return Quiver(Q.n+1,E,[-1]+list(Q.Av),Q.Ae,Q.field)

#sort edges by starting/arriving extremity
def E_in_out(Q):
    E_in=[[] for _ in range(Q.n)]
    E_out=[[] for _ in range(Q.n)]
    for i_e,e in enumerate(Q.E):
        E_in[e[1]].append(i_e)
        E_out[e[0]].append(i_e)
    return E_in,E_out


## LINEAR ALGEBRA

# Row echelon form (Gaussion pivot)
def row_echelon(M_input,field):
    M=copy.deepcopy(M_input)
    #empty matrix
    if M.shape[0]*M.shape[1]==1:
        if is_all_zero_elem([M[0][0]],field):
            return M,[0]
        return M,[]

    pivots=[]
    #create one new echelon
    def echelonify(next_pivot_row, col):
        #choose best row to pivot
        if field.descr in ['R','C']:
            best_row= next_pivot_row+np.argmax(np.abs(M[next_pivot_row:,col]))
        else:
            non_zero_rows_sub=np.where(M[next_pivot_row:,col]!=field.zero)[0]
            if len(non_zero_rows_sub)==0:
                best_row=next_pivot_row
            else:
                best_row=next_pivot_row+non_zero_rows_sub[0]
        #swap rows
        if not is_all_zero_elem([M[best_row][col]],field):
            rw=np.copy(M[next_pivot_row])
            M[next_pivot_row]=np.copy(M[best_row])
            M[best_row]=rw
            rw=np.copy(M[next_pivot_row])
            pivots.append(col)
        else: # the column col is null
            return next_pivot_row
        #echelonify the matrix
        for j, row in enumerate(M[(next_pivot_row+1):]):
          M[j+next_pivot_row+1] = row - np.array([ row[col] / rw[col] ]  )* rw
        return next_pivot_row+1

    next_pivot_row=0#nb of pivoted rows +1
    for i in range(M.shape[1]):#column to pivot
      if next_pivot_row>=M.shape[0]:#all possible rows pivoted
          break
      next_pivot_row=echelonify(next_pivot_row, i)
    #remove some computation errors
    M=flatten_zero(M,field)
    return np.array(M),pivots

#put in column echelon form
def col_echelon(M,field):
    return np.transpose(row_echelon(np.transpose(M),field)[0])


#compute kernel of M
def null_space(M,field):
    # if field is R or C: SVD
    if field.descr in ['R','C']:
        return scipy.linalg.null_space(M,rcond=epsilon)
    # otherwise column echelon of the augmanted matrix
    M=flatten_zero(M,field)
    aug_mat=flatten_zero(col_echelon(np.concatenate([M,eye_mat(M.shape[1],field)]),field),field)
    #column of the kernel base
    zero_col_top=[is_all_zero_mat(aug_mat[:M.shape[0],i], field) for i in range(M.shape[1])]
    return aug_mat[M.shape[0]:,np.array(zero_col_top)]


#intersection of two families U and V by computing the kernel of
#( U)
#(-V)
def intersection(U,V,field):
    U=flatten_zero(U,field)
    V=flatten_zero(V,field)
    M=np.concatenate((U,-V),axis=1)
    #empty matrix
    if np.shape(M)[0]*np.shape(M)[1]==0:
        return np.array([]).reshape(np.shape(M))
    u=null_space(M,field)[:np.shape(U)[1]]
    return np.dot(U,u)


# matrix of a projection from dim tot to dim b
def proj(a,b,tot,field,B=None):
    if B==None:
        B=eye_mat(b,field)
    return np.concatenate((zeros_mat(b,a,field),B,zeros_mat(b,tot-a-b,field) ),axis=1)


# transform a matrix from row echelon form to diagonal
def ech_to_diag_row(T_input,field):
    T=copy.deepcopy(T_input)
    #P_pivots s.t. T*P_pivot diag
    pivots=[]
    for i in range(min(T.shape)):
        col_piv=i
        while col_piv < T.shape[1] and is_all_zero_elem(T[i][col_piv], field) :
            col_piv+=1
        if col_piv<T.shape[1]:
            pivots.append(col_piv)
        else:
            pivots.append(-1)
    for i in range(min(T.shape)):
        if pivots[i]!=-1:
            T[i]=T[i]/T[i][pivots[i]]
    for i in range(min(T.shape)):
        if pivots[i]!=-1:
            for i_2 in range(i):
                T[i_2]= T[i_2] - np.array([T[i_2][pivots[i]]/T[i][pivots[i]]])*T[i]
    return T

# transform a matrix from column echelon form to diagonal
def ech_to_diag_col(T_input,field):
    return np.transpose(ech_to_diag_row(np.transpose(copy.deepcopy(T_input)),field))

# solve Mx=y with M triangular (square)
def solve_triangular(M,y,field):
    if y.shape[1]==0:
        return np.array([]).reshape(M.shape[0],0)

    if field.descr in ['R','C']:
        return scipy.linalg.solve_triangular(M,y)

    aug_mat=ech_to_diag_row(np.concatenate([M,y],axis=1),field)
    y_ech=aug_mat[:,M.shape[1]:]
    for i in range(M.shape[0]):
        y_ech[i]=y_ech[i]/aug_mat[i][i]
    return y_ech

## DISPLAY TOOLS

#display of a limit with its maps
def print_limit(lim_and_maps,field=Field('R')):
    lim,maps=lim_and_maps
```

```
#the limit
print("lim="+field.descr+"^"+str(lim))
#choosing a isomorphic limit with simpler maps (isomorphism given by P_pivot)
maps_prod_dim=[maps[i].shape[0]*maps[i].shape[1] for i in range(len(maps))]
i=np.argmax(np.array(maps_prod_dim))
aug_map=np.concatenate([maps[i],eye_mat(maps[i].shape[1],field)])
P_pivot=ech_to_diag_col(col_echelon(aug_map,field),field)[len(maps[i]):]
for v in range(len(maps)):
    print("lim ->",v,"\n",flatten_zero(np.dot(maps[v],P_pivot) ,field))
```

References

[BGP73] I. N. Bernstein, I. M. Gel'fand, and V. A. Ponomarev. Coxeter functors and gabriel's theorem. *USP.MAT.NAUK*, 1973.

[BJG09] Jørgen Bang-Jensen and Gregory Gutin. *Digraphs: Theory, Algorithms and Applications*, pages 198–201. Springer, 2nd edition, 2009.

[Cur14] Justin Curry. Sheaves, cosheaves and applications, 2014.

[GP08] Alasdair Graham and David Pike. A note on thresholds and connectivity in random directed graphs. *Atlantic Electronic Journal of Mathematics*, 3, 01 2008.

[Lei16] Tom Leinster. Basic category theory, 2016.

[SHN21] Anna Seigal, Heather A. Harrington, and Vidit Nanda. Principal components along quiver representations, 2021.

[TB97] Lloyd N. Trefethen and David Bau. *Numerical Linear Algebra*. SIAM, 1997.