Towards robustness in algorithms: accelerated domain decomposition, multisecant equations, and simplicial intersections

Mccoid, Conor Joseph

UNIVERSITÉ DE GENÈVE
Section de mathématiques

FACULTÉ DES SCIENCES
Martin J. Gander

Towards robustness in algorithms: accelerated domain decomposition, multisecant equations, and simplicial intersections

Ph.D. THESIS

presented to the Faculty of Science, University of Geneva
for obtaning the degree of Doctor of Mathematics.

by
Conor Joseph McCoid
from
British Columbia, Canada

Ph.D. N° 5681

Atelier d'impression ReproMail
Geneva, October 17, 2022

UNIVERSITÉ
DE GENÈVE

FACULTÉ DES SCIENCES

DOCTORAT ÈS SCIENCES, MENTION MATHÉMATIQUES

Thèse de Monsieur Conor Joseph MCCOID

intitulée :

«Towards Robustness in Algorithms: Accelerated Domain Decomposition, Multisecant Equations, and Simplicial Intersections»

La Faculté des sciences, sur le préavis de Monsieur M. GANDER, professeur ordinaire et directeur de thèse (Section de mathématiques), Monsieur G. VILMART, docteur (Section de mathématiques), Monsieur M. TRUMMER, professeur (Mathematics Department, Simon Fraser University, Bùrnaby, BC, Canada), Monsieur J. RODRIGUEZ GARCIA, professeur (Department of Applied Mathematics, Universidade de Santiago de Compostela, Santiago de Compostela, Spain), autorise l'impression de la présente thèse, sans exprimer d'opinion sur les propositions qui y sont énoncées.

Genève, le 30 août 2022

Thèse - 5681 -

[illegible]

Le Doyen

N.B. - La thèse doit porter la déclaration précédente et remplir les conditions énumérées dans les "Informations relatives aux thèses de doctorat à l'Université de Genève".

Summaries

Français

Le thèse peut diviser en trois parties: les cycles des méthodes de décomposition de domaines; l'équivalence entre les méthodes d'extrapolation et les méthodes de sous-éspace de Krylov, et; l'intersection des simplexes.

Les méthodes de décomposition de domaines subdivise un problème en tranches, et on résoud chaque tranche individuellement avant la recombinaison. La recherche dans ces méthodes concerne comment on fait cette recombinaison. Elle peut être lente, alors on veut l'accélérer, par exemple en utilisant la méthode de Newton.

Il n'existe pas de critères pour la convergence d'une méthode de décomposition de domaines accélérée par Newton. Dans le deuxième chapitre de ce thèse je présente une PDE non-linéaire pour quelle une telle méthode ne converge pas. Au lieu de convergence, cette méthode cycle entre deux pointes. Je manipule la PDE à trouver le doublement de la période et les mêmes résultats en dimension plus haut.

Pour le troisième chapitre je considère trois types de méthodes. Principalement il s'agit des méthodes d'extrapolation, mais aussi les méthodes des sous-éspace de Krylov et les équations multisécantes. Les dernières sont l'extension de la méthode sécante en dimension générale. En fait, il existe beaucoup des méthodes qui généralise cette méthode, mais toutes se rapprochent les mêmes équations. En commençant avec ces équations on peut construire les méthodes d'extrapolation et après les méthodes des sous-éspace de Krylov. Avec cette construction les trois types de méthodes devient équivalent.

Le reste du thèse concerne un algorithme d'intersection des simplexes qui est robuste. On considère premièrement l'algorithme pour les triangles, après pour les tetrahedra, et finalement pour les simplexes en dimension arbitraire. L'algorithme utilise le principe de parcimonie, qui nous dit à faire le plus petit nombre des calculations nécessaire à trouver le résultat. On enquête combien d'information chaque calcul nous informe et on ne repète pas de calculs qui nous donnent la même information.

Le plupart du dernier chapitre est sur l'implémentation de l'algorithme. Il faut se préoccuper de quelle intersections à calculer et comment on garde les calculs. Je développe les téchniques à faire ces choses dans le contexte des simplexes.

English

This thesis can be divided into three parts: cycles in domain decomposition methods; the equivalence between extrapolation methods and Krylov subspace methods, and; the intersection of simplices.

Domain decomposition methods subdivide a problem into subdomains, and resolves each subdomain individually before recombining them. Research into these methods concerns how to do the recombination procedure. It can be slow, and so we want to accelerate it, for example by using Newton's method.

There are no criteria for the convergence of a domain decomposition method accelerated by Newton's method. In the second chapter of this thesis I present a nonlinear PDE for which such a method does not converge. Instead, this method cycles between two points. By manipulating this PDE I find the period doubling and the same results in higher dimensions.

In the third chapter I consider three types of methods. Principally it concerns extrapolation methods, but also Krylov subspace methods and the multisecant equations. This latter method is an extension of the secant method to higher dimensions. In fact, there exist many methods that generalize this method, though most derive from these equations. By starting with these equations one can construct the extrapolation methods and afterwards the Krylov subspace methods. With this construction the three types of methods become equivalent.

The remainder of the thesis concerns an intersection algorithm for simplices which is robust. We consider first the algorithm for triangles, then tetrahedra, and finally simplices in arbitrary dimension. The algorithm uses the principle of parsimony, which tells us to do the smallest number of calculations necessary to find the result. We ask how much information each calculation tells us and we do not repeat calculations that tell us the same information.

Most of the last chapter is on the implementation of the algorithm. One must consider which intersections to calculate and how to store the calculations. I develop techniques to do these things in the context of simplices.

Acknowledgements

First and foremost, I must thank my supervisor, Prof. Martin J. Gander. I have been immensely lucky to have had his guidance and support during my PhD. I cannot imagine completing this work with anyone else. I also wish to thank the members of my jury for taking the time to read this thesis, ask thoughtful questions and provide crucial feedback: Prof. Manfred Trummer, Prof. Jeronimo Rodriguez Garcia, and Dr. Gilles Vilmart. This work was funded by the Swiss National Science Foundation, and I am grateful for their financial support.

I would next like to thank my coworkers at the Section of Mathematics. They were always available when help was needed, when a coffee break was called for, or when there was a serious philosophical discussion to be had. In particular, I'd like to thank my officemates Pablo Lucero and Michal Outrata. The Section saw many good people come and go during my tenure here: Adrien, Marco, Tommaso, Aitor, Liudi, Bastien, Ibrahim, Eiichi, Christine, Thibaut, Louis-Hadrien, Raphael, Renaud, Rik, Parisa, Pascaline, Giancarlo, Pratik, Ausra, Eugen, Fathi, and Elias, to name but a few. To each of you goes my thanks.

Finally, I wish to thank the friends I have made while living in the city of Geneva. Of all the things I leave behind I will miss them the most. Some have been mentioned here already, and to the rest you know who you are and what you mean to me, and so I will not sully it with words.

Contents

Chapter 1

Introduction

In which the history of this thesis is recounted.

1.1 Cycling in domain decomposition methods accelerated by Newton-Raphson

This research began as an exploration of domain decomposition methods accelerated by Newton-Raphson. Such methods can instead be thought of as Newton-Raphson preconditioned by domain decomposition methods. However, we will use the first perspective exclusively. The purpose of these methods is to solve differential equations, generally nonlinear ones, on domains that are either too complicated or too large to resolve with standard methods.

The domain is divided into subdomains and the problem solved on the subdomains. These subdomains either overlap or meet at their boundaries. The solution of a subdomain in the overlap regions or on the boundaries is shared amongst the subdomain's neighbours. The problem is resolved on the subdomains using boundary conditions defined by this shared information. This process is repeated until the solution agrees across all subdomains, called the global solution.

The iterative step, resolving the problem with new boundary conditions, can be accelerated by applying the Newton-Raphson method to it. To do so, consider the boundary conditions on each subdomain as input to a function. This function returns the boundary conditions found after solving the problem on each subdomain. The derivative of the function can be found by solving a related problem on each subdomain. Applying Newton-Raphson to this function will accelerate convergence towards the global solution, as long as the boundary conditions are near enough to those of the global solution.

This acceleration is expected to be an improvement on the convergence of domain decomposition methods, but there are no clear criteria for when it can cause divergence. The original goal was to find such a set of criteria. The focus of this research was on a specific method, namely alternating Schwarz preconditioned by Newton-Raphson (ASPN).

I began by looking for the set of criteria to guarantee convergence for Newton-Raphson when used to accelerate a fixed point iteration. This provided necessary and, in certain cases, sufficient conditions for convergence. Applying these criteria to ASPN resulted in an algorithm for guaranteed convergence of the method for nonlinear ODEs in 1D.

It also provided criteria to find counterexamples, ODEs for which ASPN converged to stable cycles. Once one was found, with a sinusoidal nonlinearity, I was able to find others.

1.2 Robust triangle intersection algorithm

I was also tasked with fixing an apparent error in an algorithm that projected information between two nonmatching triangular grids. An example was found where two overlapping triangles returned an empty intersection. After investigating I found the intersection calculation had produced inconsistent results between its subfunctions.

I then wrote a new intersection algorithm intended to keep consistency between these subfunctions. Primarily, the algorithm used a change of coordinates to keep all geometry consistent between calculations. It also created a hierarchy in the type of points that can arise in intersection calculations. Points higher in the hierarchy informed the number of points to find lower on the hierarchy.

This algorithm was submitted to Transactions on Mathematical Software, a journal published by the Association for Computing Machinery, for review. The referees pointed out that this hierarchy was not sufficient for robustness. While it ensured a given step was consistent with the preceding one, it did not provide self-consistency within that step.

To fix this, I examined the intersection calculations and found they needed to be paired together. The position of the intersections are linked to one another in groups of two. If one changes so too must another. This provided the self-consistency of the steps missing from the first version. This second version was ultimately accepted for publication.

The principles used in the triangle intersection algorithm have wider ranging applications. In particular, they can be applied to intersection algorithms for simplices in general dimension. The same hierarchy applies, and the pairing of intersections becomes a linking of a varying number of intersections.

In this general dimension the implementation becomes significantly harder. One must consider several factors, primarily how to enumerate all the calculations that need to take place.

1.3 Equivalence between extrapolation methods and Krylov methods

I next explored the equivalence between extrapolation methods and Krylov methods. Many of these equivalences were already known, and a primary goal was to codify them under a single framework.

It must be noted that Krylov methods are designed to solve linear problems. Extrapolation methods instead accelerate sequences, including nonlinear ones. When considering linear sequences the acceleration corresponds exactly to the search directions of certain Krylov methods.

Both types of methods can trace their underlying mechanisms to the multisecant equations, a set of equations that extends the secant method to higher dimensions.

These equations also underlie quasi-Newton methods, a type of root-finding method based on approximating Newton-Raphson in higher dimensions.

The final product of this research is a map showing the interconnectedness of these various methods. This map should help identify methods that can be used to solve nonlinear problems using the same techniques as Krylov methods. Since much machinery has been developed for these it would be beneficial to apply them throughout this tree of methods.

Chapter 2

ASPN

In which one examines Newton-Raphson preconditioned by alternating Schwarz as a means to discuss all methods in this family; discusses cycling behaviour in fixed point iterations, Newton and secant methods; provides a class of counter-examples where this cycling may be seen when applied to ASPN.

2.1 Introduction

Domain decomposition methods subdivide problems into subproblems. These subproblems are solved iteratively and information is passed between one another until they arrive at a solution. This iterative process can be slow and so one seeks ways to accelerate it.

Equivalently [18], one may consider these domain decomposition methods to be preconditioning some other method such as Newton-Raphson or a Krylov subspace method. This preconditioning makes the problem easier or faster to solve by splitting it into subproblems, as described above. ASPIN [9], RASPEN [13], and MSPIN [29] rely on various Schwarz methods to precondition either Newton-Raphson or inexact Newton.

Let us begin by presenting the algorithm for alternating Schwarz, one of the simplest domain decomposition methods. Suppose we seek to solve the boundary value problem

$$F(x,u,u',u'')=0,\quad x\in[a,b],\quad u(a)=A,\quad u(b)=B$$

for some function $F(x,u,v,w)$. The domain $[a,b]$ is split into two parts, $[a,\beta]$ and $[\alpha,b]$ with $\alpha<\beta$. The problem is solved on each subdomain and boundary data is passed from one subdomain to the other. One iteration of alternating Schwarz can then be summarized in the following three steps:

$$\begin{array}{llll}
(1) & F(x,u_1,u_1',u_1'')=0, & u_1(a)=A, & u_1(\beta)=\gamma_n,\\
(2) & F(x,u_2,u_2',u_2'')=0, & u_2(\alpha)=u_1(\alpha), & u_2(b)=B,\\
(3) & \gamma_{n+1}=u_2(\beta)=G(\gamma_n). & &
\end{array}$$

The function $G(\gamma)$ thus represents one iteration of alternating Schwarz in substructured form. It takes as input γ the value of $u_1(\beta)$, and returns as output the value of $u_2(\beta)$. The process is repeated until convergence, i.e.

$$(G\circ G\circ\cdots\circ G)(\gamma)=G^n(\gamma)\approx G^{n+1}(\gamma)=(G\circ G^n)(\gamma).$$

This is naturally a fixed point iteration applied to the function $G(\gamma)$.

To accelerate the method one applies Newton-Raphson to the function $f(\gamma) = G(\gamma) - \gamma$, which has a root at the fixed point. If the fixed point is unique, this is the only root of $f(\gamma)$. To apply Newton-Raphson, one needs to know the value of $G'(\gamma)$, which may be found by adding two new steps, (1') and (2'), to alternating Schwarz:

$$\begin{array}{lrll}
(1) & F(x, u_1, u_1', u_1'') = 0, & u_1(a) = A, & u_1(\beta) = \gamma_n, \\
(1') & J(u_1)\cdot(v_1, v_1', v_1'') = 0, & v_1(a) = 0, & v_1(\beta) = 1, \\
(2) & F(x, u_2, u_2', u_2'') = 0, & u_2(\alpha) = u_1(\alpha), & u_2(b) = B, \\
(2') & J(u_2)\cdot(v_2, v_2', v_2'') = 0, & v_2(\alpha) = 1, & v_2(b) = 0, \\
(3) & \gamma_{n+1} = \gamma_n - \dfrac{u_2(\beta) - \gamma_n}{v_1(\alpha)v_2(\beta) - 1} = & \gamma_n - \dfrac{G(\gamma_n) - \gamma_n}{G'(\gamma_n) - 1}, &
\end{array}$$

where $v_i(x) = \partial u_i(x)/\partial\gamma$ and $J(u_i)$ is the Jacobian of $F(x, u_i, u_i', u_i'')$.

While *a priori* convergence criteria have been found for the underlying Schwarz method, so far none exist for their combination with Newton-Raphson. We examine cycling behaviour in this accelerated alternating Schwarz method to show the difficulties with finding such criteria.

2.2 Convergence of generic fixed point iterations and Newton-Raphson

A generic fixed point iteration $x_{n+1} = g(x_n)$ converges when $|g(x_n) - x^*| < |x_n - x^*|$, where x^* is the fixed point, as this indicates $g(x_n)$ is closer to x^* than x_n. This occurs when $g(x)$ lies between x and $2x^* - x$.

The convergence or divergence of the fixed point iteration is monotonic if $\text{sign}(g(x) - x^*) = \text{sign}(x - x^*)$. In this case $g(x)$ and x lie on the same side of x^*. If this is not the case then $g(x)$ finds itself on the opposite side of the fixed point and so oscillates.

This creates four lines, $y = x$, $y = 2x^* - x$, $y = x^*$ and $x = x^*$, that divide the plane into octants. The four pairs of opposite octants form four regions with distinct behaviour of the fixed point iteration, see left of Figure 2.1 or Figure 5.7 from [21]:

1, $g(x) < x < x^*$ **or** $g(x) > x > x^*$**:** monotonic divergence;

2, $x < g(x) < x^*$ **or** $x > g(x) > x^*$**:** monotonic convergence;

3, $x < x^* < g(x) < 2x^* - x$ **or** $x > x^* > g(x) > 2x^* - x$**:** convergent oscillations;

4, $x < x^* < 2x^* - x < g(x)$ **or** $x > x^* > 2x^* - x > g(x)$**:** divergent oscillations.

If the function $g(x)$ intersects the line $y = x$ at a point other than x^* then there are additional fixed points that the method can converge towards. If it intersects the line $y = 2x^* - x$ then a stable cycle can form. A fixed point iteration is therefore only guaranteed to converge if $g(x)$ lies entirely between the lines $y = x$ and $y = 2x^* - x$, i.e. within regions 2 and 3.

Newton-Raphson can make use of this analysis by considering it as a fixed point iteration:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} = g_f(x_n).$$

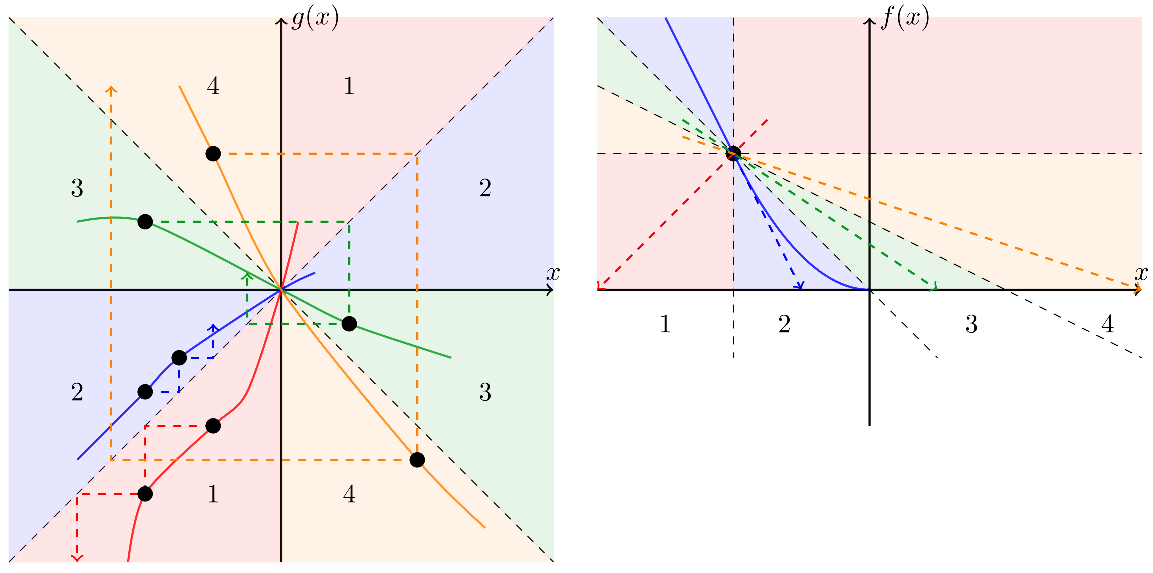


Figure 2.1: **Left:** Behaviour of the fixed point iteration $x_{n+1} = g(x_n)$, where the origin is the fixed point, $g(0) = 0$. **Right:** Regions of Newton-Raphson, $x_{n+1} = x_n - f(x_n)/f'(x_n)$, where the origin is the root, $f(0) = 0$. The tangent line to $f(x)$ can be traced from $(x, f(x))$ towards the line $y = 0$. Where it lands on this line indicates which fixed point iteration behaviour occurs.

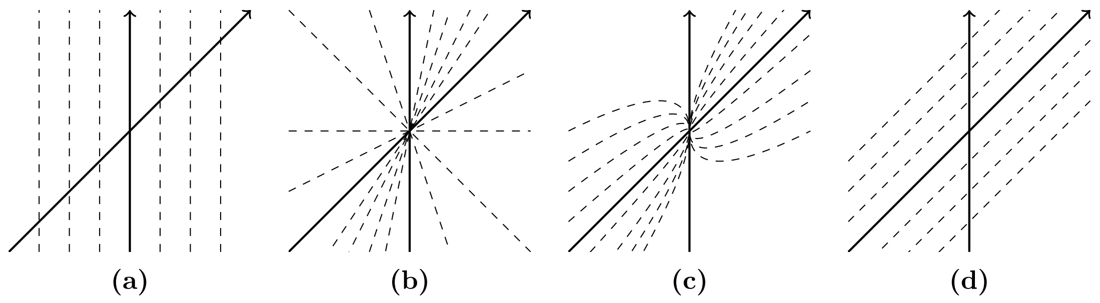


Figure 2.2: Borders between respective regions, traced from tangent lines. **(a)** Regions 1 and 2; **(b)** 2 and 3; **(c)** 3 and 4; **(d)** 4 and 1.

The borders between the regions no longer depend solely on the value of $f(x)$ but also $f'(x)$. The right of Figure 2.1 shows which type of behaviour Newton-Raphson will have based on where the tangent line points.

The borders of the regions are summarized in Figure 2.2. The slopes of most of these borders are easy to see. If $f'(x) = 0$ then $f(x)$ lies on the border between regions 1 and 4. If $f'(x) = \infty$ then it is between regions 1 and 2. If $f(x)$ is linear then it is between regions 2 and 3 and converges instantaneously.

The border between regions 3 and 4 changes depending on the value of x. Recall that this border is represented by $y = 2x^* - x$. As stated, if $g_f(x)$ intersects the line $y = x$ there are additional fixed points, and if it intersects $y = 2x^* - x$ there may be stable cycles. For guaranteed convergence $g_f(x)$ must lie between these lines. Intersections of $g_f(x)$ with $y = x$ occur only if $f(x) = 0$ and $f(x)$ has additional roots or $f'(x) = \infty$. Both circumstances are assumed not to occur. Intersections of $g_f(x)$ with $y = 2x^* - x$

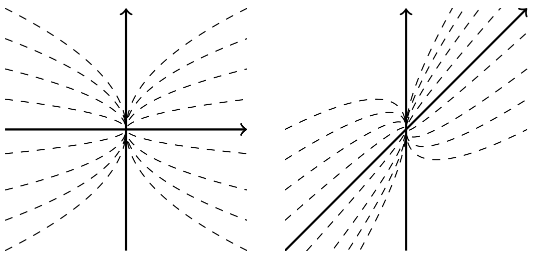


Figure 2.3: **Left:** Solutions $f_C(x)$ such that $g_f(x)$ intersects $y = 2x^* - x$ for all x. **Right:** Functions $g_C(x) = f_C(x) + x$ such that $g_f(x)$ for $f(x) = g_C(x) - x$ intersects $y = 2x^* - x$ for all x.

may be represented as a first order ODE:

$$f'_C(x) = -\frac{f_C(x)}{2(x^* - x)}, \quad f_C(x^*) = 0.$$

The solution to this ODE is $f_C(x) = C\sqrt{|x - x^*|}$ where $C \in \mathbb{R}$.

If a function $f(x)$ with root x^* is tangential to $f_C(x)$ for any value of C then $g_f(x)$ intersects the line $y = 2x^* - x$. The left of Figure 2.3 shows the functions $f_C(x)$. A function $f(x)$ that is monotonic with respect to this geometry has guaranteed convergence under Newton-Raphson. That is, if $f(x)$ is nowhere tangential to $f_C(x)$ in a given domain containing x^* for any value of C then $g_f(x)$ converges to the root for any initial guess in that domain. Since $f'_C(x^*) = \infty$ and $f(x^*) = 0$ there is always a region around the root x^* where $f(x)$ crosses all of these lines monotonically. This conforms with the theory on Newton-Raphson, which states the method has guaranteed convergence starting sufficiently near the root.

Branin suggests an adjustment to Newton-Raphson [3]:

$$x_{n+1} = x_n - \frac{f(x)}{|f'(x)|}.$$

This naturally assumes that $f(x) > 0$ when $x < x^*$ and $f(x) < 0$ when $x > x^*$. If the reverse is true, one may apply the method to $-f(x)$. Using this method requires knowledge of the position of x^* with respect to x. However, if one has this knowledge then this method may overcome many of the pitfalls of Newton-Raphson. The lines $f_C(x)$ must also incorporate the sign change for this method:

$$f_C(x) = C\,\mathrm{sign}(x - x^*)\sqrt{|x - x^*|}.$$

The corresponding geometry for a fixed point function accelerated by Newton-Raphson is skewed such that the line $y = 0$ is aligned to $y = x$, as seen in the right of Figure 2.3. The lines of this figure are the functions $g_C(x) = f_C(x) + x$. A function $g(x)$ must be monotonic in this geometry or Newton-Raphson applied to $g(x) - x$ may exhibit cycling behaviour.

The slopes of the lines $g_C(x)$ can then indicate necessary and, in some cases, sufficient conditions for Newton-Raphson to converge. To use these conditions, one must know which fixed point region of the left of Figure 2.1 the function $g(x)$ lies. One can then find the maximum and minimum slope of $g_C(x)$ in those regions. If $g(x)$ has a slope below these values then Newton-Raphson must converge, thus giving sufficient conditions. If there is not a minimum then $g'(x)$ must still adhere to the maximum, giving necessary conditions.

Table 2.1: Conditions for convergent behaviour of Newton-Raphson applied to $g(x) - x$.

$g(x)$ lies in	Necessary condition	Sufficient condition
1	$g'(x) > 1$	
2	$g'(x) < 1$	$g'(x) < 1/2$
3	$g'(x) < 1/2$	$g'(x) < 0$
4	$g'(x) < 0$	

Region 1: Given that $\sqrt{|x - x^*|}$ is non-negative, $g_C(x) < x$ only if $C < 0$ and $g_C(x) > x$ only if $C > 0$. Thus, if $g_C(x)$ lies within region 1 then $\operatorname{sign}(C) = \operatorname{sign}(x - x^*)$ and $g'_C(x) > 1$.

Region 2: The lower boundary is already defined by $C = 0$ by the above work. Likewise, by the above work $\operatorname{sign}(C) \neq \operatorname{sign}(x - x^*)$. Next, one examines where $g_C(x) < x^*$:

$$C < \frac{x^* - x}{\sqrt{|x^* - x|}} = \operatorname{sign}(x^* - x)\sqrt{|x^* - x|} \implies C^2 \begin{cases} < |x^* - x| & x^* > x, \\ > |x^* - x| & x^* < x. \end{cases}$$

One can also show that $g'_C(x)$ is bounded within this region:

$$\begin{aligned} g'_C(x) &= 1 + \operatorname{sign}(x - x^*)\frac{C}{2\sqrt{|x - x^*|}} > 1 + \frac{1}{2}\operatorname{sign}(x - x^*)\operatorname{sign}(C) = \frac{1}{2}, \\ g'_C(x) &< 1 + \operatorname{sign}(x - x^*)\lim_{x \to \pm\infty}\frac{C}{2\sqrt{|x - x^*|}} = 1 + 0 = 1, \\ &\implies 1/2 < g'_C(x) < 1. \end{aligned}$$

Region 3: As before, the lower boundary is defined by the previous region. For the upper boundary, one considers where $g_C(x) < 2x^* - x$:

$$\begin{aligned} x + C\sqrt{|x - x^*|} < 2x^* - x &\implies C < \operatorname{sign}(x^* - x)2\sqrt{|x^* - x|} \\ &\implies \frac{C^2}{4} \begin{cases} < |x^* - x| & x^* > x, \\ > |x^* - x| & x^* < x. \end{cases} \end{aligned}$$

As in region 2, one can show that $g'_C(x)$ is bounded in region 3:

$$g'_C(x) > 1 + \operatorname{sign}(x - x^*)\frac{C}{|C|} = 0.$$

Region 4: No extra work is required for this region. If $g_C(x)$ is in region 4 then $\operatorname{sign}(C) \neq \operatorname{sign}(x - x^*)$, $|x^* - x| < C^2/4$ and $g'_C(x) < 0$.

For convergence it is necessary that $g(x)$ satisfies the same conditions as $g_C(x)$. If not, then there exist values of x for which $g(x)$ is parallel to $g_C(x)$ for some value of C. It is also sufficient if $g'(x) < g'_C(x)$ for all possible C in the region which contains $g(x)$. The list of these necessary and sufficient conditions is summarized in Table 2.1.

2.2.1 Convergence of Newton-Raphson in higher dimensions

Newton's method may be generalized to higher dimensions by replacing the derivative term with the Jacobian or total derivative, denoted here by $J_f(x)$:

$$x_{n+1} = x_n - J_f(x_n)^{-1} f(x_n). \tag{2.1}$$

For $f(x)$ a vector field ($f : \mathbb{R}^n \to \mathbb{R}^n$) $J_f(x)$ is a matrix (of size $n \times n$).

One is concerned with when $\|x_{n+1} - x^*\| < \|x_n - x^*\|$, as this is where the method converges unconditionally. Divergence occurs when the reverse inequality is true. Equality is then the boundary between these regions, as in the 1D case. As was done there we search for an equation that governs when such an equality occurs.

Any vector with the same norm as $x_n - x^*$ is a rotation of $x_n - x^*$ around the origin. Thus, if R is a rotation matrix ($R^\top R = I$) then

$$\begin{aligned} x_{n+1} - x^* &= x_n - x^* - J_f(x_n)^{-1} f(x_n) \\ &= R(x_n - x^*). \end{aligned}$$

Simplifying and isolating for $f(x_n)$ we arrive at:

$$f(x_n) = J_f(x_n)(I - R)(x_n - x^*). \tag{2.2}$$

If f satisfies this equation for a point $x_n \in \mathbb{R}^n$ then that point is on the boundary between convergence and divergence. The existence of such a boundary permits (but does not guarantee) the existence of cycles. The closest boundary of this nature to the fixed point is necessarily unstable. A second boundary beyond the first may be either stable or unstable.

The equation is not readily solvable for general rotations. However, for $R = -I$ (the antipodal rotation) one can extend the result from 1D to higher dimensions. That is, the vector field:

$$f(x) = \begin{bmatrix} \sqrt{|x_1|} \\ \vdots \\ \sqrt{|x_n|} \end{bmatrix}$$

is one possible solution for $R = -I$.

Note that any particular solution immediately defines a family of solutions. Let $f(x)$ be a solution to the governing equation 2.2. Then $Af(x)$ for any matrix $A \in \mathbb{R}^{n \times n}$ is also a solution:

$$J_{Af}(x_n) = AJ_f(x_n) \implies Af(x_n) = J_{Af}(x_n)(I - R)(x_n - x^*).$$

2.3 The fixed point iteration of alternating Schwarz

We now seek to apply this theory to alternating Schwarz. As stated earlier, we consider alternating Schwarz as a function $G(\gamma)$, taking as input the value of $u_1(\beta)$ and as output the value of $u_2(\beta)$. Under reasonable conditions we can prove a number of useful properties of $G(\gamma)$ without prior knowledge of the fixed point γ^*.

Theorem 2.1. *If the problem $F(x, u, u', u'') = 0$ for $x \in \Omega$, $u(x) = h(x)$ for $x \in \partial\Omega$ has a unique solution on $\Omega = [a, \alpha]$ and $\Omega = [\beta, b]$ and the continuations of these solutions are also unique, then the function $G(\gamma)$ is strictly monotonic.*

Proof It suffices to show that $G(\gamma_1) = G(\gamma_2)$ implies $\gamma_1 = \gamma_2$. Let u_1^j solve the problem on $[a, \beta]$ with $u_1^j(\beta) = \gamma_j$. Likewise, u_2^j solves the problem on $[\alpha, b]$ with $u_2^j(\alpha) = u_1^j(\alpha)$. Suppose $u_2^1(\beta) = u_2^2(\beta)$. Then both u_2^1 and u_2^2 solve the same problem on $[\beta, b]$. By assumption, this must mean $u_2^1 = u_2^2$ and $u_1^1(\alpha) = u_1^2(\alpha)$. By a similar argument, this implies u_1^1 and u_1^2 solve the same problem on $[a, \alpha]$. Again by assumption $u_1^1 = u_1^2$ and $\gamma_1 = \gamma_2$. ■

We can even prove that $G(\gamma)$ is restricted to region 2 with additional properties. As an example, we reprove a result from Lui [30].

Theorem 2.2 (Theorem 2 from [30])**.** *Consider the equation $u''(x) + f(x, u, u') = 0$ for $x \in (a, b)$, $u(a) = u(b) = 0$ under the assumptions that*

- $f \in C^1([a, b] \times \mathbb{R} \times \mathbb{R})$,
- $\frac{\partial f(x,v,v')}{\partial u} \leq 0$ *for all* $x \in [a, b]$ *and* $v \in H_0^1([a, b])$,
- $|f(x, v, v')| \leq C(1 + |v'|^\eta)$ *for all* $x \in [a, b]$ *and* $v \in H_0^1([a, b])$ *and some* $C > 0$, $0 < \eta < 1$.

The problem is solved using alternating Schwarz with two subdomains and Dirichlet transmission conditions. Then $G(\gamma)$ for this problem lies within region 2.

Proof It suffices to prove that the problem is well posed and $0 < G'(\gamma) < 1$ for all $\gamma \in \mathbb{R}$. The well-posedness of the problem is guaranteed by Proposition 2 from [30]. As Lui points out, this also means the problem is well posed on any subdomain. Using Theorem 2.1 this gives monotonicity of $G(\gamma)$. Moreover, if $u(x) = 0$ for any $x \in (a, b)$ then the problem would be well posed on the domains $[a, x]$ and $[x, b]$. As such, $u(x)$ has the same sign as γ and $G'(\gamma) > 0$.

Consider the problem in g_1:

$$g_1''(x) + \frac{\partial f}{\partial u} g_1 + \frac{\partial f}{\partial u'} g_1' = 0, \quad x \in [a, \beta], \quad g_1(a) = 0, \quad g_1(\beta) = 1.$$

From the second assumption on f the operator on g_1 satisfies a maximum principle (see, for example, [30]). Therefore, $g_1(x) < 1$ for all $x \in (a, \beta)$. By the same reasoning, $g_2(x) < g_1(\alpha) < 1$ for all $x \in (\alpha, b)$ and $G'(\gamma) < 1$. Incidentally, the same maximum principle applies for the operator on $-g_1$ and $-g_2$, and so $G'(\gamma) > 0$ as we had before. ■

This provides guaranteed convergence of alternating Schwarz. However, it does not guarantee the convergence when one accelerates it through Newton-Raphson. Using Table 2.1 we know that such convergence is assured if $G'(\gamma) < 1/2$ for all γ, but this is not true in all cases and cannot be determined *a priori*.

Take as an example the following second order nonlinear differential equation

$$u''(x) - \sin(\mu u(x)) = 0, \quad x \in (-1, 1), \tag{2.3}$$

with homogeneous Dirichlet boundary conditions. The problem is well posed and admits only the trivial solution $u(x) = 0$. It is easy to see that this equation satisfies the conditions of Theorem 2.2. Therefore, the alternating Schwarz fixed point iteration, $G(\gamma)$, lies within region 2 and is guaranteed to converge to the fixed point. Sadly, its Newton-Raphson acceleration will not do so for all initial conditions. Take $\mu = 3.6$

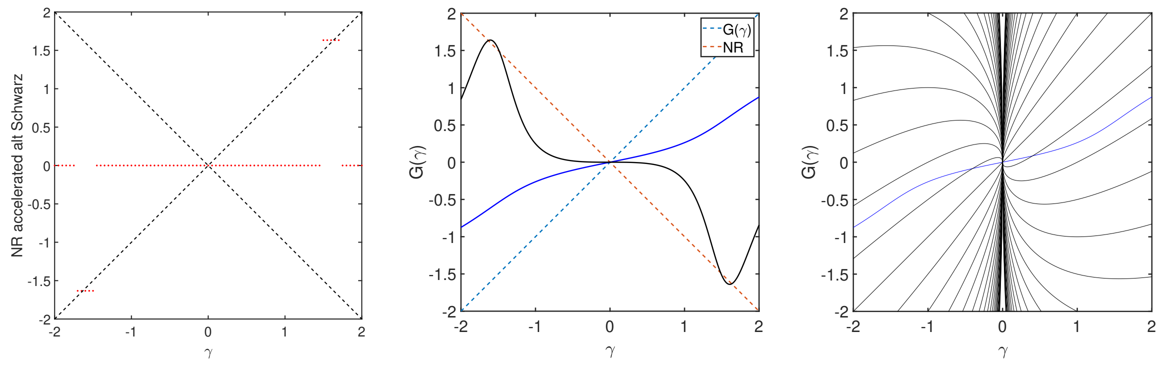


Figure 2.4: **Left:** Results of Newton-Raphson accelerated alternating Schwarz as a function of initial condition in solving equation (2.3). The value of μ is 3.6 and the subdomains are $\Omega_1 = (-1, 0.2)$ and $\Omega_2 = (-0.2, 1)$. **Middle:** $G(\gamma)$ and its Newton-Raphson acceleration. **Right:** $G(\gamma)$ plotted with the geometry of the right of Figure 2.3.

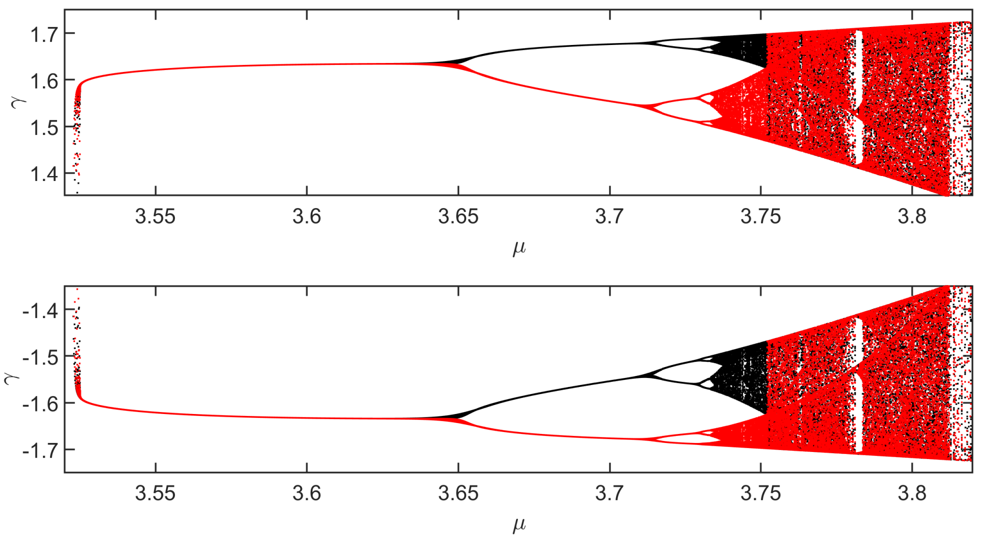


Figure 2.5: Period doubling bifurcation in the example caused by Newton-Raphson acceleration.

with an overlap of 0.4 and symmetric regions. The results of the Newton-Raphson acceleration are found in Figure 2.4 (left). While for most initial values of γ the method converges to the correct solution $u = 0$ there are two small intervals where the method enters a stable cycle.

The function $G(\gamma)$ can be plotted numerically, along with its Newton-Raphson acceleration, see Figure 2.4 (middle), which shows that $G(\gamma)$ does indeed lie within region 2 as predicted by Theorem 2.2. However, $G(\gamma)$ runs tangential to one of the lines $g_C(\gamma)$, see Figure 2.4 (right), and so its Newton-Raphson acceleration crosses into region 4. Due to symmetry, there is a 2-cycle at each crossing. Depending on the slope of the acceleration as it crosses into region 4 this cycle may be stable.

Where stable cycles exist so too must there be period doubling bifurcation. Changing the value of the parameter μ we find that the 2-cycle found in Figure 2.4 (left) becomes two 2-cycles, then two 4-cycles, and so on until it devolves into chaos, see Figure 2.5. With enough chaos the cycles are no longer stable and the acceleration

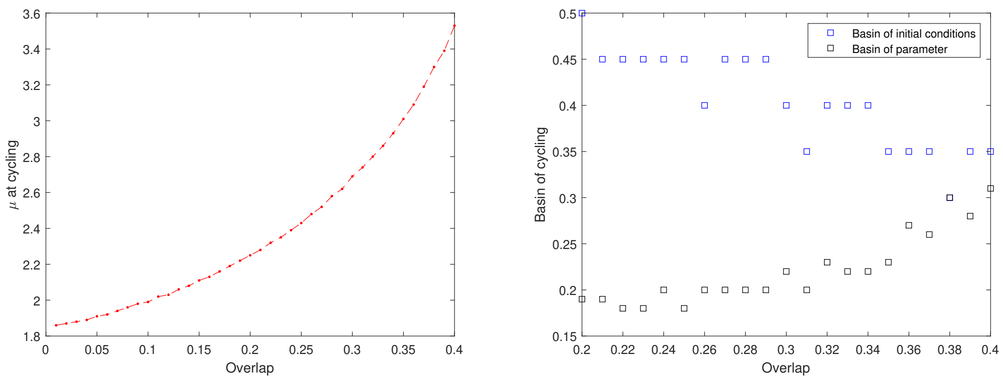


Figure 2.6: **Left:** value of μ at which bifurcation starts. **Right:** width of basin of cycling in γ and μ.

exits into a convergent region.

While a change in the parameter μ is the most obvious way to alter the dynamics, one can also change the size of the overlap. This has a direct effect on the basin of cycling in the spaces of both initial condition γ and the parameter μ. Figure 2.6 (left) shows a nonlinear relationship between the first value of μ at which cycling is observed and the size of the overlap. As the overlap grows the parameter μ must be larger and larger for cycling to occur. Figure 2.6 (right) indicates that the interval of initial conditions that result in cycling shrinks as the overlap grows. Meanwhile, the length of the bifurcation diagram increases, meaning there are more values of μ with stable cycling.

2.3.1 Alternating Schwarz and its fixed point iteration in higher dimensions

Now consider a more general alternating Schwarz, again for a second order nonlinear differential equation but on a domain $\Omega \in \mathbb{R}^d$:

$$\begin{cases} F(x, u_1^n, Du_1^n, D^2u_1^n) = 0 & x \in \Omega_1 \\ u_1^n(x) = h(x) & x \in \partial\Omega \\ u_1^n(x) = u_2^{n-1}(x) & x \in \Gamma_1 \end{cases} \begin{cases} F(x, u_2^n, Du_2^n, D^2u_2^n) = 0 & x \in \Omega_2 \\ u_2^n(x) = h(x) & x \in \partial\Omega \\ u_2^n(x) = u_1^n(x) & x \in \Gamma_2, \end{cases}$$

where $\Gamma_1 = \partial\Omega_1 \setminus \partial\Omega$ and $\Gamma_2 = \partial\Omega_2 \setminus \partial\Omega$. Note that Du (and D^2u) represents the collection of the partial derivatives (and second partial derivatives, including mixed derivatives) of the function $u(x)$.

As in the 1D case, one can construct the implicit function $G : L_2(\Gamma_1) \to L_2(\Gamma_1)$ using the following two steps:

$$(1) \begin{cases} F(x, u_1, Du_1, D^2u_1) = 0 \\ u_1(\partial\Omega) = h \\ u_1(\Gamma_1) = \gamma \end{cases} \qquad (2) \begin{cases} F(x, u_2, Du_2, D^2u_2) = 0 \\ u_2(\partial\Omega) = h \\ u_2(\Gamma_2) = u_1(\Gamma_2) \end{cases}$$

so that $G(\gamma) = u_2(\Gamma_1)$.

We are now interested in modifying the third and fourth steps of the algorithm presented for 1D so that they may be applied to higher dimensions. Let us focus on the discrete case, where $\gamma \in \mathbb{R}^N$ for some N. Likewise in the discrete case $\Gamma_1 \in \mathbb{R}^N$

and $\Gamma_2 \in \mathbb{R}^M$. Then $G(\gamma)$ is a vector field and $G'(\gamma) \in \mathbb{R}^{N\times N}$ for each γ. As well, the derivative of u_1 with respect to γ is now a Jacobian.

Let $g_1 : \Omega_1 \to \mathbb{R}^N$ represent $\frac{\partial u_1}{\partial \gamma}$. Then the third step may be written as:

$$(3) \begin{cases} J(u_1) \cdot g_1 = 0 \\ g_1(\partial\Omega) = 0 \\ g_1(\Gamma_1) = I_{N\times N} \end{cases}$$

where the last line signifies that the i–th entry of $g_1(x_j)$ is equal to $\delta_{i,j}$ (the Kronecker delta) for all $x_j \in \Gamma_1$. The object to pass to the second subdomain, $g_1(\Gamma_2)$, is then a matrix with each column representing a value of Γ_2 and each row representing a differentiation with respect to an element of γ. Thus it is of size $N \times M$.

Using the same representation for the other domain, taking $g_2 : \Omega_2 \to \mathbb{R}^M$ to be the derivative of u_2 with respect to $u_1(\Gamma_2)$, one may write the fourth step as:

$$(4) \begin{cases} J(u_2) \cdot g_2 = 0 \\ g_2(\partial\Omega) = 0 \\ g_2(\Gamma_2) = I_{M\times M}. \end{cases}$$

Then $g_2(\Gamma_1) \in \mathbb{R}^{M\times N}$ and $G'(\gamma) = g_1(\Gamma_2)g_2(\Gamma_1)$.

The fifth step is then standard high dimension Newton-Raphson:

$$\begin{aligned} (5) \quad \gamma_{n+1} &= \gamma_n - (G'(\gamma_n) - I)^{-1}(G(\gamma_n) - \gamma_n) \\ &= \gamma_n - (g_1(\Gamma_2)g_2(\Gamma_1) - I)^{-1}(u_2(\Gamma_1) - \gamma_n). \end{aligned}$$

Theorem 2.1 can be generalized to higher dimensions.

Theorem 2.3. *If the problem*

$$\begin{cases} F(u, Du, D^2u) = 0 & x \in \Omega \\ u(x) = h(x) & x \in \partial\Omega \end{cases}$$

is nonsingular on $\Omega \setminus \Omega_1$ and $\Omega \setminus \Omega_2$ in the sense that there exists a unique solution to the problem on those domains and the continuations of these solutions are also unique, then $G(\gamma)$ is injective.

Proof The proof is identical to that for theorem 2.1 with all objects replaced by their higher dimension counterparts. ■

If $G(\gamma)$ is also continuous (we require differentiable to use Newton's method) and it can be proven that alternating Schwarz converges unconditionally to a unique solution (ex. theorem 2 from [30]) then any starting point defines a path leading to the fixed point. These paths do not intersect and $G(\gamma)$ is monotonic along each path, in the sense that if the path were parametrized by a variable s then $\gamma(s_2) = G(\gamma(s_1))$ implies $s_2 > s_1$.

2.4 Accelerated alternating Schwarz with guaranteed convergence

Given Theorem 2.2 and the conditions of Table 2.1 one can construct a series of tests to see if the Newton-Raphson acceleration is suitable for a given iteration. We present one further useful trick to strengthen convergence, a correction to Newton-Raphson due to Davidenko and Branin [3, 4, 12]. We replace step (3) in the algorithm with

$$(3*) \quad \tilde{\gamma}_n = \gamma_n - \frac{G(\gamma_n) - \gamma_n}{|G'(\gamma_n) - 1|}.$$

For $G(\gamma)$ within region 2 the Newton-Raphson acceleration will now always march in the direction of the fixed point. It may still overshoot and cycle but the direction will always be correct.

For a problem satisfying the conditions of Theorem 2.2 or similar that guarantees that $G(\gamma)$ lies in region 2 the algorithm proceeds as follows.

Algorithm 2.1. *1. Select some $\gamma_0 \in \mathbb{R}$. Set $n = 0$.*

2. *Calculate $G(\gamma_n)$ and $G'(\gamma_n)$. If $G'(\gamma_n) = 1$ then set $\gamma_{n+1} = G(\gamma_n)$, increment n and return to step 2. If this is not true, proceed to step 3.*

3. *Perform step (3*), which is the Newton-Raphson acceleration using the Davidenko-Branin trick. If $|G'(\gamma_n) - 1| \geq 1/2$ then set $\gamma_{n+1} = \tilde{\gamma}_n$, increment n and return to step 2. If this is not true, calculate $\hat{\gamma}_n$, the average of γ_n and $\tilde{\gamma}_n$, and proceed to step 4.*

4. *Calculate $G(\hat{\gamma}_n)$. If $G(\hat{\gamma}_n) - \hat{\gamma}_n$ has the same sign as $G(\gamma_n) - \gamma_n$ then set $\gamma_{n+1} = \tilde{\gamma}_n$, increment n and return to step 2. If this is not true, set $\gamma_{n+1} = G(\gamma_n)$, increment n and return to step 2.*

Each of steps 2, 3 and 4 contain a test of whether Newton-Raphson will converge. In step 2, Newton-Raphson will not converge if the derivative of $G(\gamma) - 1$ is zero. In step 3, convergence is guaranteed if $G'(\gamma) \leq 1/2$ based on Table 2.1. The Davidenko-Branin trick strengthens this and also guarantees convergence if $G'(\gamma) \geq 3/2$.

In step 4 we test the point halfway between the starting value γ_n and the Newton-Raphson acceleration $\tilde{\gamma}_n$, denoted $\hat{\gamma}_n$. Since $G(\gamma)$ is in region 2 if $G(\gamma) > \gamma$ then $\gamma < \gamma^*$ and vice versa. Therefore, we can easily determine whether $\hat{\gamma}_n$ is on the same side of the fixed point as γ_n. If it is, then the fixed point γ^* lies on the same side of $\hat{\gamma}_n$ as $\tilde{\gamma}_n$, and so $\tilde{\gamma}_n$ is closer to γ^* than γ_n. If it is not, then γ^* lies between γ_n and $\hat{\gamma}_n$. Since $\tilde{\gamma}_n$ is on the other side of $\hat{\gamma}_n$ it is further from γ^* than γ_n and we have divergence. In such a case, the fixed point iteration should be used.

Note that while $G(\gamma)$ represents alternating Schwarz in this context, it may be exchanged for any fixed point iteration, in particular any Schwarz method. All that is required for the algorithm to function is for $G(\gamma)$ to be within region 2. For Schwarz methods, this would necessitate a theorem similar to Theorem 2.2.

2.5 Finding the space of counterexamples

We seek a larger space of examples where cycling occurs for ASPN. To do so, we employ optimization techniques. First we must find a functional that takes a nonlinearity and returns a measure of the chaos that results from applying ASPN.

We consider the set of problems

$$u''(x) + f(u(x)) = 0, \quad x \in (-1, 1) \tag{2.4}$$

with homogeneous Dirichlet boundary conditions. The function $f(x)$ satsifies $f(0) = 0$ so that $u(x) = 0$ is the solution to the ODE.

The counterexample already presented makes use of the antisymmetry in $G(\gamma)$ to achieve its cycles. We wish the same for all nonlinearities $f(x)$ in our space of counterexamples.

Proposition 2.4. *If $f(x)$ is antisymmetric then $G(\gamma)$ is antisymmetric.*

Proof Suppose $\hat{u}_1(x)$ solves step (1) of alternating Schwarz with $u_1(\beta) = \gamma$. Then

$$-\hat{u}_1'' + f(-\hat{u}_1) = f(\hat{u}_1) - f(\hat{u}_1) = 0.$$

Thus, $-\hat{u}_1(x)$ solves step (1) of alternating Schwarz with $u_1(\beta) = -\gamma$.

By the same logic, if $\hat{u}_2(x)$ solves step (2) of alternating Schwarz with $u_2(\alpha) = u_1(\alpha)$ then $-\hat{u}_2(x)$ solves step (2) with $u_2(\alpha) = -u_1(\alpha)$. Thus, $-\hat{u}_2(\beta) = -G(\gamma) = G(-\gamma)$. ■

We therefore restrict our search to nonlinearities which are antisymmetric. It is then sufficient for the Newton-Raphson acceleration, represented by $G_N(\gamma)$, to cross the line $y = -\gamma$ for cycles to exist.

Given that $f(x)$ is antisymmetric it can be decomposed into a Fourier series consisting solely of sinusoids:

$$f(x) = \sum_{k=1}^{N} c_k \sin(\pi k x). \tag{2.5}$$

Thus, the functional to optimize takes a set $\{c_k\}_{k=1}^N \in \mathbb{R}^N$, passes it through $f(x) \in C(-1,1)$ and $G(\gamma) \in C(\mathbb{R})$ to arrive at a measure for the chaos of the system. We will represent this functional as $L : \mathbb{R}^N \to \mathbb{R}$. There are many ways to define $L(\{c_k\})$, but we shall use

$$L(\{c_k\}) := \|G_N(\gamma) + \gamma\|. \tag{2.6}$$

Thus, $L(\{c_k\}) = 0$ if and only if $G_N(\gamma) = -\gamma$, and ASPN cycles between γ and $-\gamma$ for all values of γ.

It is well-known that there always exists a region around the root of a function where Newton-Raphson converges, assuming continuity of $G''(\gamma)$ and $G'(\gamma) \neq 1$. We expect then to find only local minimizers of $L(\{c_k\})$. We use a gradient descent with line search to optimize the functional L, with restrictions $\gamma \in [-2, 2]$ and $\{c_k\}_{k=1}^5 \in \mathbb{R}^N$, with starting condition $c_1 = 1$, $c_i = 0$ for $i = 2, \ldots, 5$. There is an overlap of 0.4 between the domains, which are symmetric about $x = 0$. We find an exceptional counterexample using this methodology, as presented in Figure 2.7.

We now seek counterexamples in 2D. That is, we seek $f : \mathbb{R} \to \mathbb{R}$ such that using ASPN to solve

$$u_{xx}(x,y) + u_{yy}(x,y) + f(u(x,y)) = 0, \quad x, y \in (-1,1)$$

with homogeneous Dirichlet boundary conditions results in cycling behaviour. The two domains are split along the x-axis, so that the first domain is $x \in (-1, \alpha)$, $y \in (-1,1)$ and the second is $x \in (-\alpha, 1)$, $y \in (-1,1)$.

As before, $f(x)$ is chosen to be antisymmetric so that $G(\gamma)$ is antisymmetric. Proposition 2.4 applies to the higher dimensional case by replacing all relevant scalar objects (γ, $G(\gamma)$) with their corresponding vectors ($\boldsymbol{\gamma}$, $\boldsymbol{G}(\boldsymbol{\gamma})$). The decomposition of $f(x)$ into sinusoids and the definition of the functional $L(\{c_k\})$ remain unchanged.

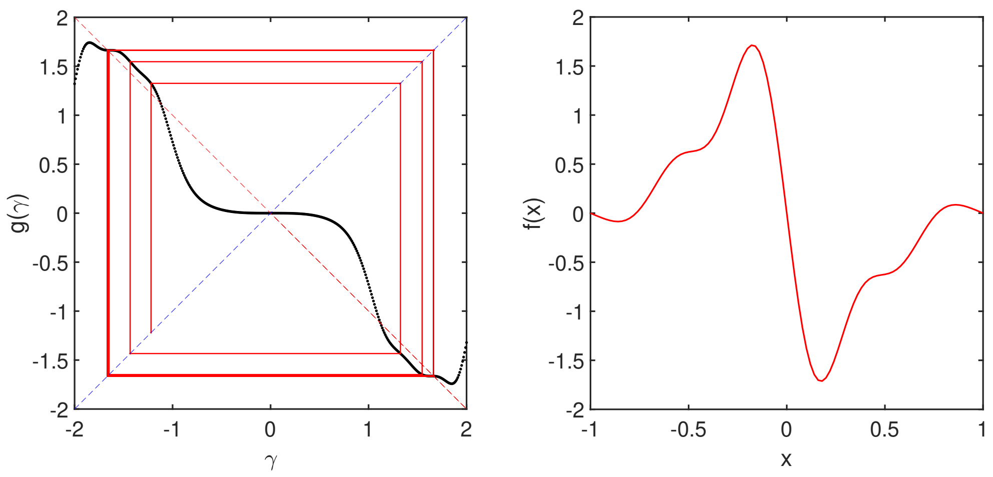


Figure 2.7: A counterexample found through optimizing the functional L, equation (2.6). (Left) ASPN falls into a stable 2-cycle; the basin of attraction of this cycle is most values within $[-2,-1) \cup (1,2]$. (Right) The function $f(x)$ for this counterexample; it is the sum of five sinusoids.

If γ is the discretization of a sinusoid then $G(\gamma) = c\gamma$, where $c \in \mathbb{R}$, up to numerical error. This can be seen by transforming the solution into a Fourier series in the y variable. Suppose

$$\begin{cases} \frac{\partial^2}{\partial x^2} u_1(x,y) + \frac{\partial^2}{\partial y^2} u_1(x,y) + f(u_1(x,y)) = 0, \quad x \in (-1,\alpha), \quad y \in (-1,1) \\ u_1(-1,y) = u_1(x,\pm 1) = 0 \\ u_1(\alpha,y) = C\sin(\pi m y) \end{cases}$$

where $f(x)$ is antisymmetric and can therefore be expressed in the form of equation (2.5). Use as an ansatz $u_1(x,y) = g(x)\sin(\pi m y)$. Take the Fourier transform of the equation:

$$\begin{aligned} 0 =& \int_{-1}^{1} (g''(x) - m^2\pi^2 g(x))\sin(\pi m y)\sin(\pi k y) + f(g(x)\sin(\pi m y))\sin(\pi k y)dy \\ =&(g''(x) - m^2\pi^2 g(x))\delta_{m,k}C_m + \int_{-1}^{1}\sum_{j=1}^{N} c_j \sin(\pi j g(x)\sin(\pi m y))\sin(\pi k y)dy \\ =&\delta_{m,k}\mathcal{L}g(x) + \sum_{j=1}^{N} c_j \int_{-1}^{1}\sum_{n=0}^{\infty}\frac{(\pi j g(x))^{2n+1}}{(2n+1)!}\sin(\pi m y)^{2n+1}\sin(\pi k y)dy \\ =&\delta_{m,k}\mathcal{L}g(x) + \sum_{j=1}^{N} c_j \sum_{n=0}^{\infty}\frac{(\pi j g(x))^{2n+1}}{(2n+1)!}S(2n+1), \end{aligned}$$

where $S(n)$ is the integral of $\sin(\pi m y)^n \sin(\pi k y)$ over $(-1,1)$. We can find a recursive formula for $S(2n+1)$:

$$\begin{aligned}
S(2n+1) =& -\frac{1}{\pi k}\cos(\pi k y)\sin(\pi m y)^{2n+1}\Big|_{-1}^{1} \\
&+ \int_{-1}^{1}\frac{m}{k}(2n+1)\cos(\pi k y)\cos(\pi m y)\sin(\pi m y)^{2n}dy \\
=&(2n+1)\frac{m}{k}\Bigg(-\frac{1}{\pi k}\cos(\pi m y)\sin(\pi m y)^{2n}\sin(\pi k y)\Big|_{-1}^{1} \\
&+\frac{m}{k}2n\int_{-1}^{1}\sin(\pi k y)\left(-\sin(\pi m y)^{2n+1}+\cos(m\pi y)^2\sin(\pi m y)^{2n-1}\right)dy\Bigg) \\
=&(2n+1)(2n)\frac{m^2}{k^2}\int_{-1}^{1}\sin(\pi m y)^{2n-1}\sin(\pi k y)-2\sin(\pi m y)^{2n+1}\sin(\pi k y)dy \\
=&(2n+1)(2n)\frac{m^2}{k^2}\left(S(2n-1)-2S(2n+1)\right) \\
=&\frac{(2n+1)(2n)\frac{m^2}{k^2}}{1+2(2n+1)(2n)\frac{m^2}{k^2}}S(2n-1) \\
=&C(m,k,n)S(1).
\end{aligned}$$

The value of $S(1)$ is zero unless $k=m$. This proves there exists a function $\tilde{f}_m(x)$ such that

$$\mathcal{L}g(x)+\tilde{f}_m(g(x))=0.$$

Thus, if $g(x)$ satisfies this ODE with boundary conditions $g(-1)=0$ and $g(\alpha)=C$ then $u_1(x,y)$ solves the PDE. Since $u_1(x,y)$ in this form satisfies the boundary conditions it is the unique solution to this step of alternating Schwarz. The solution on the second domain, $u_2(x,y)$, then also has the same form by a symmetry argument, and its value at $x=-\alpha$ is a sinusoid of the same period. Therefore, $G(\gamma)=c\gamma$.

As a direct consequence of this, starting with any single sine wave as boundary conditions provides a single parameter pathway for the function $G:\mathbb{R}^N\to\mathbb{R}^N$. We take advantage of this fact and use $c\sin(\pi y)$ as the boundary condition for the first subdomain of ASPN, varying c between -0.5 and 0. To seed the optimization we use the previously obtained counterexample nonlinearity $f(x)$ from the 1D case. The same optimization method is used.

The resulting nonlinearity does not admit a stable cycle but highlights the chaos that can result from using ASPN. Figure 2.8 gives this nonlinearity and an example of a sequence generated by ASPN. The sequence begins on the sine wave path described previously. It then approaches a nearby stable cycle, having departed from the strict sine wave path to one that closely resembles sine waves (bottom left of figure). However, the cycle is not numerically stable and is ultimately abandoned. As it leaves this pathway it descends into a divergent regime, with the norm growing exponentially (top left). It quickly ejects onto a convergent pathway (top right). It is possible these cycles exist on saddlepoints, stable along some pathways but unstable along others. A small numerical error will then shunt the ASPN sequence away from the cycle, either to a divergent (top left) or convergent (top right) pathway.

Figure 2.9 provides snapshots of the solution at each of the iterates. The first 24 iterations of ASPN are shown, giving the nearly cycling regime (iterations 1 to 12),

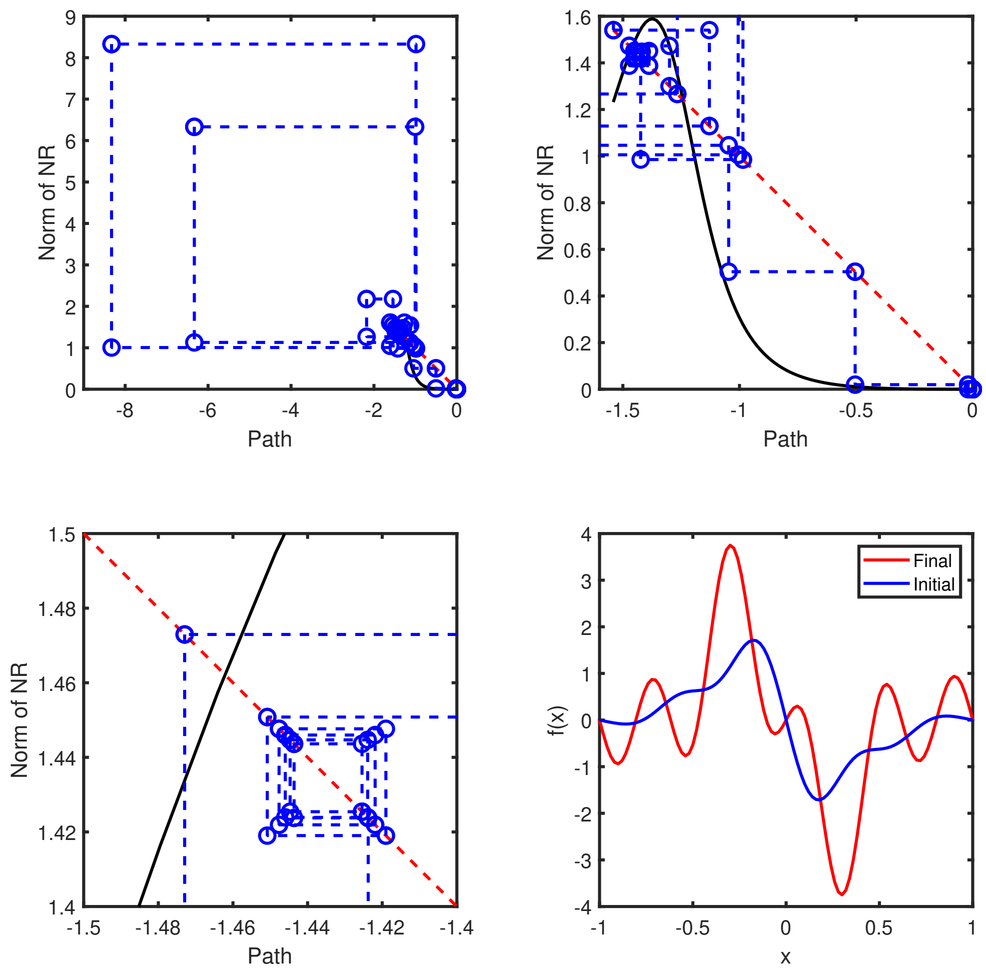


Figure 2.8: A chaotic ASPN sequence in 2D, at three resolutions (top row, bottom left). The nonlinearity (bottom right) is found through optimization on the functional L, equation (2.6). The ASPN sequence is seeded using a sine wave $\sin(\pi y)$ as boundary condition on the first subdomain, but quickly diverges from this pathway.

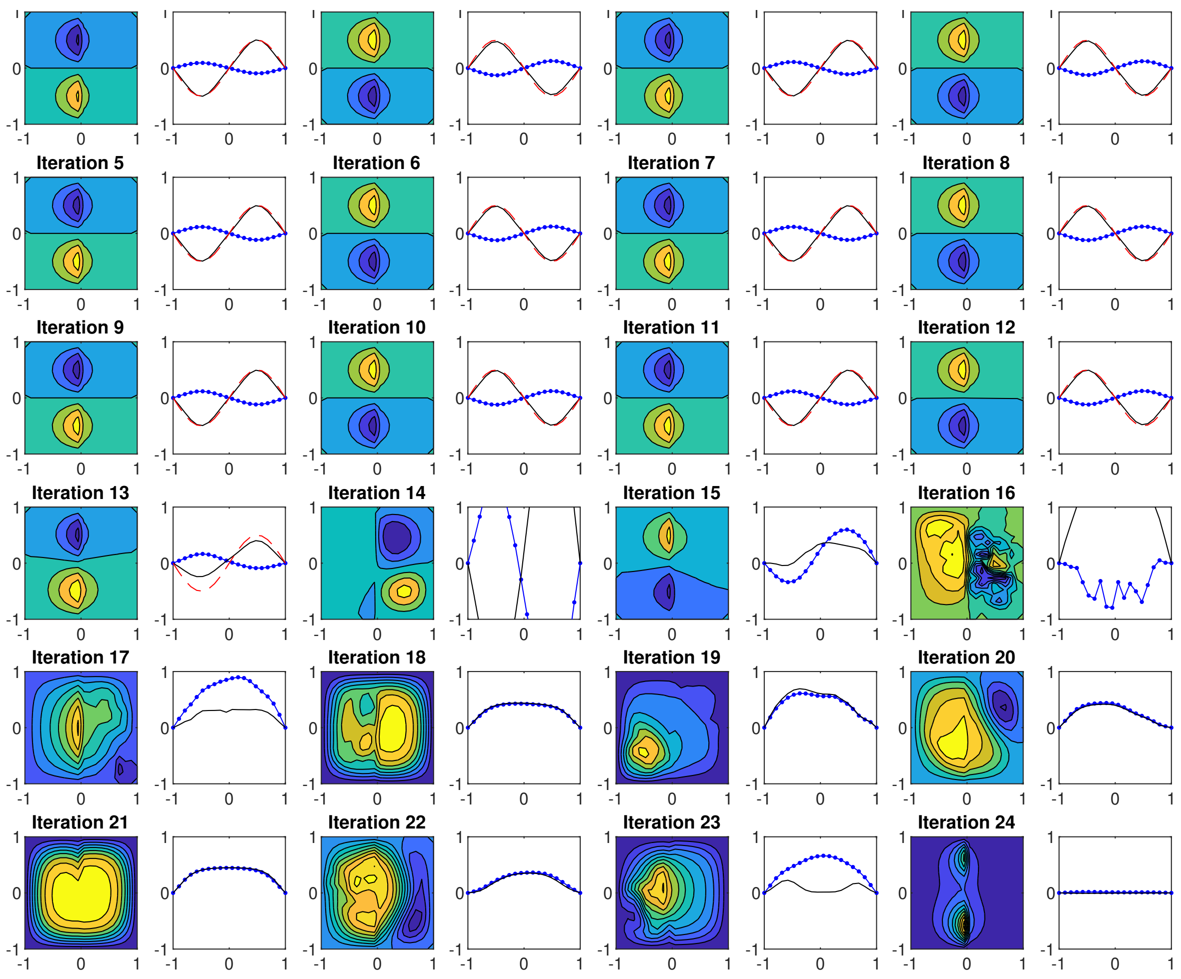


Figure 2.9: The first 24 iterations of the chaotic ASPN sequence. The left figure of each iteration shows the overall solution at that step combined from the two subdomains, and the right of each shows the resulting $G(\gamma)$ in blue and the ASPN result in black. For comparison, the sine wave $0.5\sin(\pi y)$ is plotted for the cycling regime, with sign that alternates with each iteration.

the divergent regime (13 to 20) and the convergent regime (21 to 24). The cycling regime is very nearly a cycle of sine waves, as seen by the comparable sine waves in red. A small change from these sine waves in iteration 13 causes chaos to take over until relative stability in iteration 20. From there, ASPN converges quickly, arriving at the solution by iteration 24. The remaining iterates resolve this solution to higher precision.

2.6 Conclusions

These results show that acceleration cannot be used without consequences for all Schwarz algorithms. As with standard Newton-Raphson, there exist problems for which the sequence diverges, cycles or behaves chaotically.

In 1D, necessary and sufficient conditions for convergence could be found, assuming certain properties of the problem being solved. Using these conditions as well as known tricks to stabilize Newton-Raphson, one can construct a convergent algorithm. Higher dimensions proves more challenging, as more can go wrong.

Starting from a single counterexample we found many more through optimization of a functional which increases when the iterates become cyclical. This work was extended into 2D, showing that adding dimensionality does not resolve the problem.

Chapter 3

Multisecant equations

In which one connects multisecant equations with extrapolation methods, Krylov subspace methods and Broyden's family; discusses the many possible ways in which one can solve the multisecant equations, with comparison of efficiency and accuracy.

3.1 Introduction

The equivalence between extrapolation methods and Krylov subspace methods is well-studied [37, 38, 27]. These have largely focused on individual extrapolation methods and the orthogonalization processes involved in each. Equivalence between Krylov subspace methods and quasi-Newton methods, including the multisecant equations, is less studied but still known [44]. Gragg and Stewart describe using a QR factorization to solve the multisecant equations [24]; If the function evaluations form a Krylov subspace this would be exactly GMRES [35].

Sidi [39] has developed a framework for extrapolation methods, while Fang and Saad [16] have developed one for quasi-Newton methods, but neither includes consideration of the other type of methods, nor Krylov subspace methods. While the framework presented here does not include every method considered in both of these previous frameworks, it shows all three types of methods connect fundamentally.

3.2 Multisecant equations

The multisecant equations are a generalization of the secant method into higher dimensions. Recall that the secant method seeks the root of the function $f(x)$ by computing an approximation of the derivative $f'(x_n)$:

$$\hat{x} = x_{n+1} - (x_{n+1} - x_n)\left(f(x_{n+1}) - f(x_n)\right)^{-1} f(x_{n+1}).$$

In higher dimensions it is necessary to find an approximation to the Jacobian $J(\mathbf{x}_n)$. To do so, one can expand the function $\mathbf{f}(\mathbf{x})$ into a Taylor series about $\mathbf{x}_n$:

$$\mathbf{f}(\mathbf{x}_{n+i}) = \mathbf{f}(\mathbf{x}_n) + J(\mathbf{x}_n)(\mathbf{x}_{n+i} - \mathbf{x}_n) + \frac{1}{2}(\mathbf{x}_{n+i} - \mathbf{x}_n)H(\mathbf{x}_n)(\mathbf{x}_{n+i} - \mathbf{x}_n) + \ldots$$

As a first order approximation we can take the first two terms of this series, resulting in the following approximate equation [2]:

$$\mathbf{f}(\mathbf{x}_{n+i}) - \mathbf{f}(\mathbf{x}_n) \approx J(\mathbf{x}_n)(\mathbf{x}_{n+i} - \mathbf{x}_n).$$

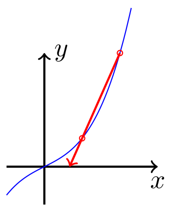


Figure 3.1: Example of the secant method. Two points $(x_1, f(x_1))$ and $(x_2, f(x_2))$ are used to draw a line. The zero of this line is then used as the next estimate of the root of $f(x)$.

This system can be solved for $J(\mathbf{x}_n)$, though such a system would be underdetermined. However, if one had as many $\mathbf{f}(\mathbf{x}_{n+i})$ as there are dimensions in the space then one could solve

$$\begin{bmatrix}\mathbf{f}(\mathbf{x}_{n+1}) & \dots & \mathbf{f}(\mathbf{x}_{n+d})\end{bmatrix} - \mathbf{f}(\mathbf{x}_n)\mathbf{1}^\top = \hat{J}\left(\begin{bmatrix}\mathbf{x}_{n+1} & \dots & \mathbf{x}_{n+d}\end{bmatrix} - \mathbf{x}_n\mathbf{1}^\top\right) \tag{3.1}$$

where $\hat{J}$ approximates $J(\mathbf{x}_n)$. This system is nonsingular given sufficient conditions on the choice of the $\mathbf{x}_{n+i}$.

Now that an approximation $\hat{J}$ has been found for the Jacobian, an approximate root may be calculated:

$$\hat{\mathbf{x}} = \mathbf{x}_n - \hat{J}^{-1}\mathbf{f}(\mathbf{x}_n).$$

This system is referred to as the multisecant equations. They can be further generalized by allowing $\hat{J}$ to be the solution of an underdetermined system, such as by multiplying both sides of equation (3.1) by a matrix $B^\top$.

The multisecant equations are but one instance of several equivalent methods. We begin with a specific form of the equations and prove the general form. Let

$$F_{n,k} = \begin{bmatrix}\mathbf{f}(\mathbf{x}_n) & \dots & \mathbf{f}(\mathbf{x}_{n+k})\end{bmatrix}, \quad X_{n,k} = \begin{bmatrix}\mathbf{x}_n & \dots & \mathbf{x}_{n+k}\end{bmatrix}, \quad \Delta_n = \begin{bmatrix}-1 & \dots & -1 \\ 1 & & \\ & \ddots & \\ & & 1\end{bmatrix} \tag{3.2}$$

then equation (3.1) may be rewritten as

$$\hat{J}^{-1}F_{n,k}\Delta_n = X_{n,k}\Delta_n.$$

We require the vector $\hat{J}^{-1}\mathbf{f}(\mathbf{x}_n)$. If there exists a vector $\tilde{\mathbf{u}}$ such that $F_{n,k}\Delta_n\tilde{\mathbf{u}} = \mathbf{f}(\mathbf{x}_n)$ then $\hat{J}^{-1}\mathbf{f}(\mathbf{x}_n) = X_{n,k}\Delta_n\tilde{\mathbf{u}}$. Thus, the multisecant equations can be represented in the following compact form:

$$F_{n,k}\Delta_n\tilde{\mathbf{u}} = \mathbf{f}(\mathbf{x}_n), \quad \hat{\mathbf{x}} = \mathbf{x}_n - X_{n,k}\Delta_n\tilde{\mathbf{u}}. \tag{3.3}$$

If there are fewer function evaluations than the dimension of the space, $k < d$, the equation to solve $\hat{J}$ is underdetermined. In this instance, the first system of equation (3.3) is overdetermined. There are two options to then solve this system: either to pad out the matrix $F_{n,k}$ with additional vectors, or add constraints. We will focus on the latter, but will discuss the former in Section 3.3. Adding constraints results in the underdetermined Newtonian form of the multisecant equations:

$$B^\top F_{n,k}\Delta_n\tilde{\mathbf{u}} = B^\top\mathbf{f}(\mathbf{x}_n), \quad \hat{\mathbf{x}} = \mathbf{x}_n - X_{n,k}\Delta_n\tilde{\mathbf{u}}. \tag{3.4}$$

One can replace $\Delta_n \tilde{\mathbf{u}}$ with $\hat{\mathbf{u}}$, transforming equation (3.4) into

$$\begin{bmatrix} \mathbf{1}^\top \\ B^\top F_{n,k} \end{bmatrix} \hat{\mathbf{u}} = \begin{bmatrix} 0 \\ B^\top \mathbf{f}(\mathbf{x}_n) \end{bmatrix}, \quad \hat{\mathbf{x}} = \mathbf{x}_n - X_{n,k} \hat{\mathbf{u}}. \tag{3.5}$$

The additional constraint $\mathbf{1}^\top \hat{\mathbf{u}} = 0$, that the elements sum to 0, ensures a bijection between $\tilde{\mathbf{u}}$ and $\hat{\mathbf{u}}$.

Proposition 3.1. *For all $\hat{\mathbf{u}} \in \mathbb{R}^k$ there exists a unique $\tilde{\mathbf{u}} \in \mathbb{R}^{k+1}$ such that $\hat{\mathbf{u}} = \Delta \tilde{\mathbf{u}}$ where $\mathbf{1}^\top \hat{\mathbf{u}} = 0$, the columns of Δ sum to zero, and its first k rows form an invertible matrix.*

Proof Using the constraint $\mathbf{1}^\top \hat{\mathbf{u}} = 0$ one can write $\hat{\mathbf{u}}_{k+1} = -\mathbf{1}^\top R_k \hat{\mathbf{u}}$, where R_k is the restriction operator that takes the first k elements of a vector of length $k+1$. Then $\tilde{\mathbf{u}}$ can be found by solving the system $R_k \Delta \tilde{\mathbf{u}} = R_k \hat{\mathbf{u}}$. The matrix $R_k \Delta$ is square and invertible, meaning $R_k \hat{\mathbf{u}}$ uniquely determines $\tilde{\mathbf{u}}$. Since $R_k \hat{\mathbf{u}}$ also uniquely determines $\hat{\mathbf{u}}_{k+1}$, there exists exactly one $\tilde{\mathbf{u}}$ for any given $\hat{\mathbf{u}}$. ■

One can then replace $\hat{\mathbf{u}}$ with $\mathbf{e}_1 - \mathbf{u}$:

$$\begin{aligned} \begin{bmatrix} \mathbf{1}^\top \\ B^\top F_{n,k} \end{bmatrix} (\mathbf{e}_1 - \mathbf{u}) &= \begin{bmatrix} 0 \\ B^\top \mathbf{f}(\mathbf{x}_n) \end{bmatrix}, & \hat{\mathbf{x}} =& \mathbf{x}_n - X_{n,k}(\mathbf{e}_1 - \mathbf{u}) \\ \begin{bmatrix} 1 \\ B^\top \mathbf{f}(\mathbf{x}_n) \end{bmatrix} - \begin{bmatrix} \mathbf{1}^\top \\ B^\top F_{n,k} \end{bmatrix} \mathbf{u} &= & =& \mathbf{x}_n - \mathbf{x}_n + X_{n,k} \mathbf{u}. \end{aligned}$$

Rearranging gives the base form of the multisecant equations [47]:

$$\begin{bmatrix} \mathbf{1}^\top \\ B^\top F_{n,k} \end{bmatrix} \mathbf{u} = \begin{bmatrix} 1 \\ \mathbf{0} \end{bmatrix}, \quad \hat{\mathbf{x}} = X_{n,k} \mathbf{u}. \tag{3.6}$$

From this base form one can transform into several equivalent forms. One interesting example is to use the transform $\mathbf{u} = \hat{\mathbf{u}} - \mathbf{e}_i$, giving

$$\begin{bmatrix} \mathbf{1}^\top \\ B^\top F_{n,k} \end{bmatrix} \hat{\mathbf{u}} = \begin{bmatrix} 0 \\ B^\top \mathbf{f}(\mathbf{x}_{n+i}) \end{bmatrix}, \quad \hat{\mathbf{x}} = \mathbf{x}_{n+i} - X_{n,k} \hat{\mathbf{u}}.$$

One can then replace $\hat{\mathbf{u}}$ with $\Delta \tilde{\mathbf{u}}$ for any $\Delta \in \mathbb{R}^{k+1 \times k}$ such that its columns sum to zero:

$$F_{n,k} \Delta \tilde{\mathbf{u}} = \mathbf{f}(\mathbf{x}_{n+i}), \quad \hat{\mathbf{x}} = \mathbf{x}_{n+i} - X_{n,k} \Delta \tilde{\mathbf{u}}.$$

Our original Δ_n is a natural choice, but one can also use

$$\Delta_s = \begin{bmatrix} -1 & & \\ 1 & \ddots & \\ & \ddots & -1 \\ & & 1 \end{bmatrix}.$$

Since all three forms are equivalent for all choices of i and valid Δ and the multisecant equations represents a specific choice of i and Δ, all three forms provide the same approximation to the root of $\mathbf{f}(\mathbf{x})$ as the multisecant equations. Wherever the multisecant equations are used one may replace them with any of the forms presented here.

The multisecant equations are an example of a quasi-Newton method. A quasi-Newton method is any method of the form

$$\hat{\mathbf{x}}_{n+1} = \mathbf{x}_n - \mathbf{u}_n \tag{3.7}$$

where $\mathbf{u}_n$ is an approximate solution to the equation

$$J(\mathbf{x}_n)\mathbf{u} = \mathbf{f}(\mathbf{x}_n) \tag{3.8}$$

where $J(\mathbf{x})$ is the Jacobian of $\mathbf{f}(\mathbf{x})$ evaluated at $\mathbf{x}_n$. In particular, one can use the multisecant equations in any of their forms to provide such an approximation.

Consider, for example, equation (3.3) with $k = d$. The solution $\tilde{\mathbf{u}}$ may be found elementwise by Cramer's rule:

$$\begin{aligned}\tilde{\mathbf{u}}_i =& \frac{\begin{vmatrix} \dots & \mathbf{f}(\mathbf{x}_{n+i-1}) - \mathbf{f}(\mathbf{x}_n) & \mathbf{f}(\mathbf{x}_n) & \mathbf{f}(\mathbf{x}_{n+i+1}) - \mathbf{f}(\mathbf{x}_n) & \dots \end{vmatrix}}{\begin{vmatrix} \mathbf{f}(\mathbf{x}_{n+1}) - \mathbf{f}(\mathbf{x}_n) & \dots & \mathbf{f}(\mathbf{x}_{n+d}) - \mathbf{f}(\mathbf{x}_n) \end{vmatrix}} \\ =& (-1)^i \frac{\begin{vmatrix} \mathbf{f}(\mathbf{x}_n) & \dots & \mathbf{f}(\mathbf{x}_{n+i-1}) & \mathbf{f}(\mathbf{x}_{n+i+1}) & \mathbf{f}(\mathbf{x}_{n+d}) \end{vmatrix}}{\begin{vmatrix} 1 & \dots & 1 \\ \mathbf{f}(\mathbf{x}_n) & \dots & \mathbf{f}(\mathbf{x}_{n+d}) \end{vmatrix}}.\end{aligned}$$

The quasi-Newton method defined above may then be expressed as

$$\begin{aligned}\hat{\mathbf{x}}_{n+1} =& \mathbf{x}_n - \frac{\begin{vmatrix} 0 & \mathbf{x}_{n+1} - \mathbf{x}_n & \dots & \mathbf{x}_{n+d} - \mathbf{x}_n \\ \mathbf{f}(\mathbf{x}_n) & \mathbf{f}(\mathbf{x}_{n+1}) & \dots & \mathbf{f}(\mathbf{x}_{n+d}) \end{vmatrix}}{\begin{vmatrix} 1 & \dots & 1 \\ \mathbf{f}(\mathbf{x}_n) & \dots & \mathbf{f}(\mathbf{x}_{n+d}) \end{vmatrix}} \\ =& \frac{\begin{vmatrix} \mathbf{x}_n & \dots & \mathbf{x}_{n+d} \\ \mathbf{f}(\mathbf{x}_n) & \dots & \mathbf{f}(\mathbf{x}_{n+d}) \end{vmatrix}}{\begin{vmatrix} 1 & \dots & 1 \\ \mathbf{f}(\mathbf{x}_n) & \dots & \mathbf{f}(\mathbf{x}_{n+d}) \end{vmatrix}}\end{aligned}$$

where one must expand the determinant along the top row to maintain the correct dimensions.

Suppose that we do not have enough values of $\mathbf{f}(\mathbf{x}_{n+i})$ to fully determine $\hat{J}$, i.e. $k < d$. We apply the solution found above to the underdetermined Newtonian form of the equations, equation (3.4). The quasi-Newton method that results from this may be written as

$$\hat{\mathbf{x}}_{n+1} = \frac{\begin{vmatrix} \mathbf{x}_n & \dots & \mathbf{x}_{n+k} \\ \mathbf{v}_1^\top \mathbf{f}(\mathbf{x}_n) & \dots & \mathbf{v}_1^\top \mathbf{f}(\mathbf{x}_{n+k}) \\ \vdots & & \vdots \\ \mathbf{v}_k^\top \mathbf{f}(\mathbf{x}_n) & \dots & \mathbf{v}_k^\top \mathbf{f}(\mathbf{x}_{n+k}) \end{vmatrix}}{\begin{vmatrix} 1 & \dots & 1 \\ \mathbf{v}_1^\top \mathbf{f}(\mathbf{x}_n) & \dots & \mathbf{v}_1^\top \mathbf{f}(\mathbf{x}_{n+k}) \\ \vdots & & \vdots \\ \mathbf{v}_k^\top \mathbf{f}(\mathbf{x}_n) & \dots & \mathbf{v}_k^\top \mathbf{f}(\mathbf{x}_{n+k}) \end{vmatrix}} \tag{3.9}$$

where $\mathbf{v}_i$ is the i–th column of B.

3.3 Connection to root-finding methods

As mentioned in the previous section, as an alternative to adding constraints to equation (3.1) one can add vectors to the matrix $F_{n,k}$. For example, the update for Broyden's method [7] is chosen such that

$$\hat{J}_{n+1}\begin{bmatrix}X_{n,1}\Delta & Q\end{bmatrix} = \begin{bmatrix}F_{n,1}\Delta & \hat{J}_n Q\end{bmatrix}$$

where $Q^\top X_{n,1}\Delta = 0$. The columns of Q are additional search directions, and the product $\hat{J}_n Q$ an approximation to function evaluations in these search directions. In the generalized Broyden's method [43, 15] $X_{n,1}$ and $F_{n,1}$ are replaced by $X_{n,k}$ and $F_{n,k}$ and Q reduced in size by k columns.

Broyden's family of methods [15, 16] may be written as

$$\hat{J}_{n+1} = \hat{J}_n + \mathbf{f}(\mathbf{x}_{n+1})\mathbf{v}_n^\top$$

where $\mathbf{v}_n^\top(\mathbf{x}_{n+1} - \mathbf{x}_n) = 1$. If $\mathbf{v}_n$ is chosen such that

$$\mathbf{v}_n^\top X_{n-k,k}\Delta = \begin{bmatrix}0 & \dots & 0 & 1\end{bmatrix}$$

with possibly other constraints then $\hat{J}_{n+1}X_{n-k,k}\Delta = F_{n-k,k}\Delta$.

Anderson mixing [1, 15, 44] solves $F_{n,k}\Delta\mathbf{u} = \mathbf{f}(\mathbf{x}_{n+k})$ in a least-squares sense then uses the step

$$\hat{\mathbf{x}}_{n+k+1} = \mathbf{x}_{n+k} - X_{n,k}\Delta\mathbf{u} + \beta\left(\mathbf{f}(\mathbf{x}_{n+k}) - F_{n,k}\Delta\mathbf{u}\right).$$

For $\beta = 0$ this is exactly the multisecant equations.

3.4 Connection to extrapolation methods

Extrapolation methods seek to accelerate the convergence of sequences. In general, these sequences are nonlinear and can lie within a vector space. If one has $k+1$ iterates of the sequence, $\{\mathbf{x}_n, \dots, \mathbf{x}_{n+k}\}$, then the next element in the accelerated sequence is

$$\hat{\mathbf{x}}_{n+1} = \sum_{i=0}^{k}\mathbf{u}_i\mathbf{x}_{n+i} = X_{n,k}\mathbf{u}$$

where $\mathbf{1}^\top\mathbf{u} = 1$. We choose $\mathbf{u}$ such that

$$\lim_{n\to\infty}\mathbf{x}_n = \mathbf{x} = X_{n,k}\mathbf{u} \iff (X_{n,k} - \mathbf{x}\mathbf{1}^\top)\mathbf{u} = 0.$$

The iterate $\hat{\mathbf{x}}_{n+1}$ may be expressed as

$$\hat{\mathbf{x}}_{n+1} = \frac{\begin{vmatrix}\mathbf{x}_n & \dots & \mathbf{x}_{n+k}\\ \mathbf{v}_1^\top\mathbf{r}(\mathbf{x}_n) & \dots & \mathbf{v}_1^\top\mathbf{r}(\mathbf{x}_{n+k})\\ \vdots & & \vdots\\ \mathbf{v}_k^\top\mathbf{r}(\mathbf{x}_n) & \dots & \mathbf{v}_k^\top\mathbf{r}(\mathbf{x}_{n+k})\end{vmatrix}}{\begin{vmatrix}1 & \dots & 1\\ \mathbf{v}_1^\top\mathbf{r}(\mathbf{x}_n) & \dots & \mathbf{v}_1^\top\mathbf{r}(\mathbf{x}_{n+k})\\ \vdots & & \vdots\\ \mathbf{v}_k^\top\mathbf{r}(\mathbf{x}_n) & \dots & \mathbf{v}_k^\top\mathbf{r}(\mathbf{x}_{n+k})\end{vmatrix}}$$

where $\mathbf{r}(\mathbf{x}_{n+i}) = \mathbf{x}_{n+i+1} - \mathbf{x}_{n+i}$ and $\{\mathbf{v}_i\}$ is a linearly independent set of vectors. Note this is exactly equation (3.9) replacing $\mathbf{f}(\mathbf{x})$ with $\mathbf{r}(\mathbf{x})$. Thus, extrapolation methods of this form are identical to the multisecant equations acting on $\mathbf{f}(\mathbf{x}) = \mathbf{r}(\mathbf{x})$ using $\{\mathbf{x}_{n+i}\}$ as search directions. They can therefore be expressed in the form of equation (3.6).

Methods of this form are called polynomial extrapolation algorithms [39, 27]. Several extrapolation methods fall in this category:

- minimum polynomial extrapolation (MPE) [8], with $\mathbf{v}_i = \mathbf{r}(\mathbf{x}_{n+i-1})$;
- modified minimum polynomial extrapolation (MMPE) [39], with $\mathbf{v}_i$ some fixed vector;
- reduced rank extrapolation (RRE) [14, 32, 41], with $\mathbf{v}_i = \mathbf{r}(\mathbf{x}_{n+i}) - \mathbf{r}(\mathbf{x}_{n+i-1})$.

There is another category of methods known as ϵ-algorithms [49, 22, 5, 27]. They may be connected to the multisecant equations as well, though not in the same manner as the others. Recall equation (3.1). One can take several such equations and arrive at several approximations of the Jacobian:

$$F_{n+j,k}\Delta_n = \hat{J}_{n+j}X_{n+j,k}\Delta_n.$$

As before, we seek to solve

$$F_{n+j,k}\Delta\tilde{\mathbf{u}}_j = \mathbf{f}(\mathbf{x}_{n+j}), \quad \hat{\mathbf{x}}_{n+1} = \mathbf{x}_{n+j} - X_{n+j,k}\Delta\tilde{\mathbf{u}}_j.$$

This gives several estimates of $\hat{\mathbf{x}}_{n+1}$. Each of these systems can be reduced to a single equation by taking the inner product with a given vector $\mathbf{v}$. There are then as many equations as there are values of j. If one assumes all $\tilde{\mathbf{u}}_j$ are equal then one can summarize these equations in the following system:

$$\begin{bmatrix} \mathbf{v}^\top F_{n,k} \\ \vdots \\ \mathbf{v}^\top F_{n+k-1,k} \end{bmatrix} \Delta\tilde{\mathbf{u}} = \begin{bmatrix} \mathbf{v}^\top \mathbf{f}(\mathbf{x}_n) \\ \vdots \\ \mathbf{v}^\top \mathbf{f}(\mathbf{x}_{n+k-1}) \end{bmatrix}, \quad \hat{\mathbf{x}}_{n+1} = \mathbf{x}_n - X_{n,k}\Delta\tilde{\mathbf{u}}.$$

Following the same work as in Section 3.2 one arrives at the solution

$$\hat{\mathbf{x}}_{n+1} = \frac{\begin{vmatrix} \mathbf{x}_n & \dots & \mathbf{x}_{n+k} \\ \mathbf{v}^\top\mathbf{f}(\mathbf{x}_n) & \dots & \mathbf{v}^\top\mathbf{f}(\mathbf{x}_{n+k}) \\ \vdots & & \vdots \\ \mathbf{v}^\top\mathbf{f}(\mathbf{x}_{n+k-1}) & \dots & \mathbf{v}^\top\mathbf{f}(\mathbf{x}_{n+2k-1}) \end{vmatrix}}{\begin{vmatrix} 1 & \dots & 1 \\ \mathbf{v}^\top\mathbf{f}(\mathbf{x}_n) & \dots & \mathbf{v}^\top\mathbf{f}(\mathbf{x}_{n+k}) \\ \vdots & & \vdots \\ \mathbf{v}^\top\mathbf{f}(\mathbf{x}_{n+k-1}) & \dots & \mathbf{v}^\top\mathbf{f}(\mathbf{x}_{n+2k-1}) \end{vmatrix}}.$$

Replacing $\mathbf{f}(\mathbf{x})$ with $\mathbf{r}(\mathbf{x})$ gives the topological ϵ-algorithm (TEA) [6].

3.5 Connection to Krylov methods

Suppose the extrapolation methods of the previous section are applied to the linear vector sequence $\mathbf{x}_{n+1} = (A+I)\mathbf{x}_n - \mathbf{b}$. The limit $\mathbf{x}$ of this sequence is then the solution to $A\mathbf{x} = \mathbf{b}$. The functions $\mathbf{r}(\mathbf{x}_{n+i})$ are then

$$\mathbf{r}(\mathbf{x}_{n+i}) = (A + I)\mathbf{x}_{n+i} - \mathbf{b} - \mathbf{x}_{n+i} = A\mathbf{x}_{n+i} - \mathbf{b}$$

and they satisfy

$$\mathbf{r}(\mathbf{x}_{n+i}) = (A + I)\mathbf{x}_{n+i} - \mathbf{b} - (A + I)\mathbf{x}_{n+i-1} + \mathbf{b} = (A + I)\mathbf{r}(\mathbf{x}_{n+i-1}).$$

The vectors $\mathbf{r}(\mathbf{x}_{n+i})$ then form a Krylov subspace, such that $\mathbf{r}(\mathbf{x}_{n+i}) \in \mathcal{K}_i(A+I, \mathbf{r}(\mathbf{x}_n))$.

Under these conditions the extrapolation methods become Krylov subspace methods. Most notably, since MPE uses the Arnoldi iteration to produce orthogonal search directions, it is identical to GMRES when applied to this linear sequence [44, 37, 27]. These algorithms are presented as Algorithms 3.1 and 3.2. The connections to the linear case are noted in the latter. Since $\mathcal{K}_k(A+I, \mathbf{v}) = \mathcal{K}_k(A, \mathbf{v})$ the search directions for both algorithms are identical.

Algorithm 3.1 (GMRES)**.**

```
$\mathbf{q}_1 = \mathbf{b}/\|\mathbf{b}\|$
***for*** $k = 1$ *to* $n$ ***do***
    $\mathbf{y} = A\mathbf{q}_k$ *(*$\in \mathcal{K}_k(A, \mathbf{b})$*)*
    *orthogonalize* $\mathbf{y}$ *with respect to* $\mathcal{K}_{k-1}(A, \mathbf{b})$
    $\mathbf{q}_{k+1} = \mathbf{y}/\|\mathbf{y}\|$
***end for***
*minimize* $\|H_n\mathbf{u} - \|\mathbf{b}\|\,\mathbf{e}_1\|$
$\hat{\mathbf{x}}_{n+1} = Q_n\mathbf{u} + \mathbf{x}_0$
```

Algorithm 3.2 (MPE)**.**

```
$\mathbf{q}_1 = \mathbf{r}(\mathbf{x}_{n+1})/\|\mathbf{r}(\mathbf{x}_{n+1})\|$ *(*$\mathbf{b}/\|\mathbf{b}\|$ *in linear case)*
***for*** $k = 1$ *to* $n$ ***do***
    $\mathbf{y} = \mathbf{r}(\mathbf{x}_{n+k})$ *(*$\in \mathcal{K}_k(A + I, \mathbf{b})$ *in linear case)*
    *orthogonalize* $\mathbf{y}$ *with respect to* $\{\mathbf{r}(\mathbf{x}_n), \ldots, \mathbf{r}(\mathbf{x}_{n+k-1})\}$ *(*$\mathcal{K}_{k-1}(A + I, \mathbf{b})$ *in linear
*case)*
    $\mathbf{q}_{k+1} = \mathbf{y}/\|\mathbf{y}\|$
***end for***
*minimize* $\|H_n\mathbf{u}\|$ *such that* $\mathbf{1}^\top\mathbf{u} = 1$
$\hat{\mathbf{x}}_{n+1} = \begin{bmatrix}\mathbf{x}_0 & \ldots & \mathbf{x}_n\end{bmatrix}\mathbf{u}$
```

The minimization steps of the two algorithms can be shown to be equivalent as well. Consider the solution found using GMRES: $\hat{\mathbf{x}} = Q_k\mathbf{y}_k$ where Q_k is derived from the Arnoldi iteration on $\mathcal{K}_{k-1}(A + I, \mathbf{b})$. Then we seek

$$\begin{aligned}\min\|A\hat{\mathbf{x}} - \mathbf{b}\| &= \min\|AQ_k\mathbf{y}_k - \mathbf{b}\|\\ &= \min\|F_{n,k}\Delta\tilde{\mathbf{u}}_k - \mathbf{b}\|\end{aligned}$$

since the column space of $F_{n,k}\Delta$ is equal to the Krylov subspace $\mathcal{K}_{k-1}(A, \mathbf{b})$. Recall in the linear case that $\mathbf{f}(\mathbf{x}_n) = \mathbf{b}$, and so this minimization is equivalent to solving equation (3.4) with $B = F_{n,k}$. We've shown in Section 3.2 that this is equivalent to minimizing $\|F_{n,k}\mathbf{u}_k\|$ under the constraint $\mathbf{1}^\top\mathbf{u}_k = 1$. This is exactly the minimization step in MPE.

For the linear case of TEA, the term $\mathbf{v}^\top\mathbf{f}(\mathbf{x}_{n+i})$ may now be written as $\mathbf{v}^\top(A + I)^{i-j}\mathbf{f}(\mathbf{x}_{n+j})$. This means TEA can now be expressed in the form of equation (3.6)

Extrapolation	$\mathbf{q}_{k+1}\perp$	Krylov
MPE	$\mathcal{K}_k(A+I, \mathbf{f}(\mathbf{x}_n))$	GMRES
RRE	$\mathcal{K}_k(A, \mathbf{f}(\mathbf{x}_n))$	GMRES
MMPE	$\mathcal{K}_k(G, \mathbf{q}_0)$	n/a
TEA	$\mathcal{K}_k(A^\top, \mathbf{q})$	BiCG

Table 3.1: Connections between extrapolation methods and Krylov methods. The extrapolation methods are applied to the sequence $\mathbf{x}_{n+1} = (A+I)\mathbf{x}_n - \mathbf{b}$.

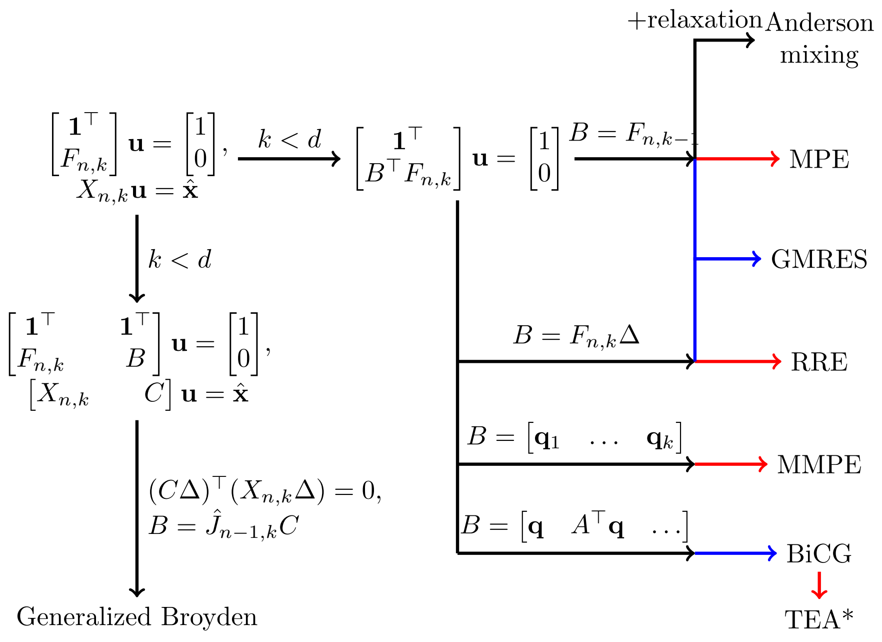

+relaxation
Anderson mixing
$\begin{bmatrix}\mathbf{1}^\top \\ F_{n,k}\end{bmatrix}\mathbf{u} = \begin{bmatrix}1\\0\end{bmatrix}$, $X_{n,k}\mathbf{u} = \hat{\mathbf{x}}$
$k<d$
$\begin{bmatrix}\mathbf{1}^\top \\ B^\top F_{n,k}\end{bmatrix}\mathbf{u} = \begin{bmatrix}1\\0\end{bmatrix}$
$B = F_{n,k-1}$
MPE
GMRES
$k<d$
$\begin{bmatrix}\mathbf{1}^\top & \mathbf{1}^\top \\ F_{n,k} & B\end{bmatrix}\mathbf{u} = \begin{bmatrix}1\\0\end{bmatrix}$, $\begin{bmatrix}X_{n,k} & C\end{bmatrix}\mathbf{u} = \hat{\mathbf{x}}$
$B = F_{n,k}\Delta$
RRE
$B = \begin{bmatrix}\mathbf{q}_1 & \dots & \mathbf{q}_k\end{bmatrix}$
MMPE
$(C\Delta)^\top(X_{n,k}\Delta) = 0$, $B = \hat{J}_{n-1,k}C$
$B = \begin{bmatrix}\mathbf{q} & A^\top\mathbf{q} & \dots\end{bmatrix}$
BiCG
Generalized Broyden
TEA*

Figure 3.2: Interconnectivity of extrapolation, acceleration and quasi-Newton methods. Red arrows indicate $\mathbf{f}(\mathbf{x}_n) = \mathbf{x}_{n+1} - \mathbf{x}_n$ while blue arrows indicate $\mathbf{f}(\mathbf{x}_n) = A\mathbf{x}_n - \mathbf{b}$ and $\mathbf{f}(\mathbf{x}_{n+1}) = (A+I)\mathbf{f}(\mathbf{x}_n)$. Note that TEA* is the linear version of the method; general TEA derives directly from the multisecant equations.

with $\mathbf{v}_i = (A^\top + I)^i\mathbf{v}$. TEA then uses the Lanczos biorthogonalization process, making it equivalent in the linear case to biorthogonal conjugate gradient (BiCG).

Table 3.1 gives the orthogonalization conditions and corresponding Krylov subspace methods for these four extrapolation methods when they are applied to linear vector sequences.

3.6 Conclusions

The multisecant equations form the basis of numerical methods for root-finding in higher dimensions. They themselves are extensions of the secant method in 1D.

When underdetermined they can be connected to extrapolation methods. Which particular extrapolation method depends on the additional constraints imposed on the underdetermined systems.

If these methods are applied to linear problems with vector sequences that lie within Krylov subspaces, then they become Krylov subspace methods. This is due to the orthogonalization steps in these methods.

The three methods are then intrinsically linked. As a direct consequence, extrapolation methods can be thought of as nonlinear Krylov subspace methods, as they use the same orthogonalization processes but applied to nonlinear problems. Moreover, this indicates the advanced techniques employed in Krylov subspace methods can be used to improve extrapolation methods. It also suggests that any Krylov subspace method has an extrapolation method counterpart, and vice versa.

Chapter 4

Intersection of triangles

In which one presents a robust algorithm for the intersection of triangles.

4.1 Introduction

This chapter deals with an algorithm I developed for the intersection of two triangles as part of a larger advancing front algorithm that calculated the projection from one triangular mesh to another. The algorithm was published in [31]. The impetus for this algorithm was a counterexample to an earlier intersection algorithm found by Jorge Albella Martínez, at the time a PhD student of Jerónimo Rodriguez Garcia from the Universidade de Santiago de Compostela. This previous algorithm, PANG [20, 19], failed to correctly identify a valid intersection between two nearly coincidental triangles.

This counterexample is presented in Figure 4.1. The algorithm identifies two vertices of the blue triangle as being inside the red triangle (apex and left base). As well, PANG identifies the left base vertex of the red triangle as being inside the blue triangle, as the two left base vertices are coincident. However, PANG then calculates only one intersection: the left base vertex of both triangles. As only two points are found for the intersection, no volume is calculated. The right of Figure 4.1 shows that a slight perturbation in all vertices of the small triangle leads to the correct result. To allow replication of the example of Figure 4.1 we present in Table 4.1 the exact positions in double precision of the triangle vertices.

One can determine the point of failure in this example by examining how PANG finds vertices of the polygon of intersection. For this task PANG uses two sub-algorithms: PointsOfXInY (which determines if a vertex of triangle U is inside triangle V) and EdgeIntersections (which calculates the intersections between the edges of two triangles). These routines use different, mathematically equivalent vector projections that, due to round-off error from floating-point arithmetic, can produce inconsistent results.

Such an inconsistency may be found in the example of Figure 4.1. According to

`Na`	-2.134346793664016	-1.294592514989478	-1.774411380685791
	0.633829593520260	0.622483665789736	1.328933577453528
`Nb`	-2.134346793664016	-0.454838236314940	-1.414475967707566
	0.633829593520260	0.611137738059212	2.024037561386796

Table 4.1: Positions of the triangle vertices of Figure 4.1.

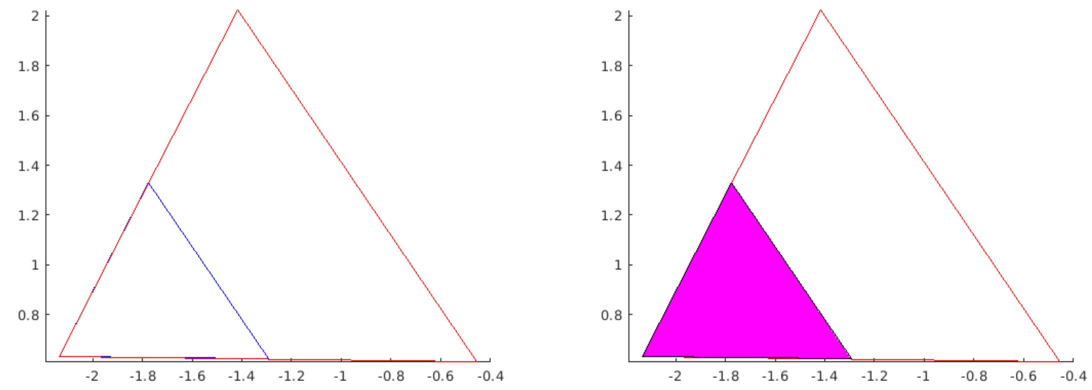


Figure 4.1: Left: example of a triangle intersection where an intersection is not calculated despite one of substantial size clearly existing. Right: perturbing vertices of the smaller triangle allows the algorithm to find the correct result. Pink shading indicates the calculated intersection.

PointsOfXInY in floating-point arithmetic, the right base vertex of the blue triangle does not lie within the red triangle. According to EdgeIntersections in floating-point arithmetic, all three vertices of the blue triangle lie inside the red triangle and so no intersections can be calculated (excepting the vertex that is coincident with a vertex of the red triangle). In short, the algorithm fails because the subroutines do not agree on whether one vertex of one triangle lies inside the other. It is clear from this example that consistency is required amongst the subroutines of the algorithm: All components must produce consistent results even in the presence of round-off error [36].

We then want to develop an intersection algorithm that provides this level of robustness to the calculation of the intersection. That is, the error in this calculation is bounded and continuous with respect to perturbations. To achieve this the algorithm enforces consistency across its numerical calculations.

4.2 Review of other algorithms

Calculating the intersection of two polygons is well-studied as the polygon clipping problem, primarily in the field of computer graphics where one wishes to know when a given 'subject' polygon U is hidden from an observer by a second 'clipping' polygon V. Speed is valued over accuracy in these applications and glitches of varying severity are commonplace in video games and computer-generated imagery, see Figure 4.2 for an example.

A number of algorithms have been proposed for the intersection of polygons. One may classify them by considering how they handle each of the three types of vertices found in the polygon of intersection: vertices of U lying inside V; the intersections between the edges of U and those of V; and the vertices of V lying inside U.

Some algorithms divide the plane into sections based on the edges of V [42, 33, 34, 28]. Each edge defines an infinite reference line of which the edge is a finite interval. This line then defines a parameter that is positive on one side and negative on the other. If all parameters associated with all edges are positive for a vertex of U then it lies inside V.

An alternative approach is to consider the projection of vectors between vertices and

Figure 4.2: Example of clipping in the game NBA 2K13 [48].

edges and the respective normal vectors [11, 40]. The goal is to find a vector projection whose sign indicates whether a vertex is in V. The calculation of an intersection would then reduce to finding where the projection is zero.

A third option is to trace out the polygon of intersection [25, 46]. One computes all intersections between all edges of U and V, then chooses an intersection and marches along the vertices of U, adding each to the polygon of intersection. Once this trace reaches another intersection between U and V it switches to march along the vertices of V. This is repeated until the first intersection is reached. Such a procedure encounters problems when U and V share vertices [20].

One can reduce the cost of calculating the intersections between U and V by first determining which edges intersect. This can be done on an edge-by-edge basis [42, 40, 28] or considering the polygons as a whole [33, 25].

In computer graphics, the polygon clipping problem often reduces to a line clipping problem [11, 40, 28]. As such, vertices of V lying inside U are often irrelevant. These can be dealt with by repeating the process for vertices of U inside V, swapping U and V. Algorithms using a trace procedure [25, 46] make no distinction between U and V and find such vertices in the same manner as those of U in V.

The Sutherland-Hodgman algorithm [42] has a unique method for finding vertices of V in U. The algorithm takes a given edge of V and finds the corresponding reference line, defining a positive side of the line which contains V and a negative side which does not. It then discards all vertices of U on the negative side of the line and calculates intersections with the line for each edge of U that had one of its vertices removed in this way. The result is a new polygon lying entirely on one side of the reference line. The process is repeated with a new edge of V until all edges of V have been used. In this way, a vertex of V in U is the last of a sequence of intersections of the edges of intermediary polygons with reference lines extending from edges of V.

The algorithm found in Section 4.5 is a successor to PANG, and therefore named PANG2. This algorithm makes use of reference lines to determine which vertices of U lie inside V, like some of the algorithms cited above [42, 33, 34, 28]. This information is then used to identify which edges intersect, similar to [42, 40, 28]. The intersection points then indicate which vertices of V lie inside U. As each step uses all available data from previous steps the algorithm is parsimonious, which is sufficient for robustness [17].

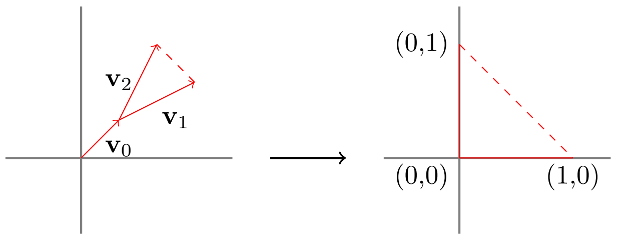


Figure 4.3: Coordinate transformation.

4.3 Change of coordinates

The triangle intersection algorithm in PANG [20] implicitly uses a change of coordinates to test whether a vertex of U, the 'subject' triangle, is inside V, the 'clipping' triangle. Effectively, the coordinates of all vertices are changed so that all calculations involving the clipping triangle are as simplified as possible. For PANG2, this change of coordinates is made explicit and used much more extensively.

We first codify the position and shape of the triangle V. Select a vertex of V. Its position is labelled by the vector $\mathbf{v}_0$. The vectors pointing from this vertex to the other two vertices of V are represented by $\mathbf{v}_1$ and $\mathbf{v}_2$, so that the positions of the vertices of V may be summarized by the matrix $\mathbf{v}_0\mathbf{1}^\top + \begin{bmatrix}\mathbf{0} & \mathbf{v}_1 & \mathbf{v}_2\end{bmatrix}$, where $\mathbf{1}$ is a column vector containing three ones and $\mathbf{0}$ a column vector containing two zeros. The same is done for U, resulting in the positions $\mathbf{u}_0\mathbf{1}^\top + \begin{bmatrix}\mathbf{0} & \mathbf{u}_1 & \mathbf{u}_2\end{bmatrix}$.

We perform a change of coordinates through an affine transformation on both V and U. The goal is to transform V into a right angle triangle aligned with the coordinate axes, called the reference triangle Y. This reference triangle has vertices at $(0,0)$, $(0,1)$ and $(1,0)$. It is clear then that the affine transformation $A\mathbf{v} + \mathbf{b}$ satisfies the equation

$$A\left(\mathbf{v}_0\mathbf{1}^\top + \begin{bmatrix}\mathbf{0} & \mathbf{v}_1 & \mathbf{v}_2\end{bmatrix}\right) + \mathbf{b}\mathbf{1}^\top = \begin{bmatrix}0 & 1 & 0\\ 0 & 0 & 1\end{bmatrix}.$$

It can be deduced that $\mathbf{b} = -A\mathbf{v}_0$ and $A\begin{bmatrix}\mathbf{v}_1 & \mathbf{v}_2\end{bmatrix} = I$, the identity matrix. Thus, A is the inverse of $\begin{bmatrix}\mathbf{v}_1 & \mathbf{v}_2\end{bmatrix}$.

The transformation of U, called X, may then be obtained by calculating $A(\mathbf{u}_0\mathbf{1}^\top + \begin{bmatrix}\mathbf{0} & \mathbf{u}_1 & \mathbf{u}_2\end{bmatrix} - \mathbf{v}_0\mathbf{1}^\top)$. Alternatively, since A is used exclusively in this step, one may solve the following system for the coordinates of the vertices of X:

$$\begin{bmatrix}\mathbf{v}_1 & \mathbf{v}_2\end{bmatrix}\begin{bmatrix}x_1 & x_2 & x_3\\ y_1 & y_2 & y_3\end{bmatrix} = \mathbf{u}_0\mathbf{1}^\top + \begin{bmatrix}\mathbf{0} & \mathbf{u}_1 & \mathbf{u}_2\end{bmatrix} - \mathbf{v}_0\mathbf{1}^\top.$$

Affine transformations are numerically stable if the matrix involved in the transformation is well-conditioned. The condition number of A is proportional to the aspect ratio of the triangle V: As the columns of A^{-1}, $\mathbf{v}_1$ and $\mathbf{v}_2$, become parallel the angle between them shrinks, thus making V thinner [26]. There are two affine transformations used in this algorithm which are inverses of each other.

The original PANG intersection algorithm performs this procedure twice: once to find vertices of U lying inside V and a second time for vertices of V lying inside U, resulting in two different affine transformations. An unrelated procedure is used to calculate the intersections. To achieve our goal of a robust algorithm, PANG2 keeps the underlying geometry of all calculations consistent: The coordinates created at this

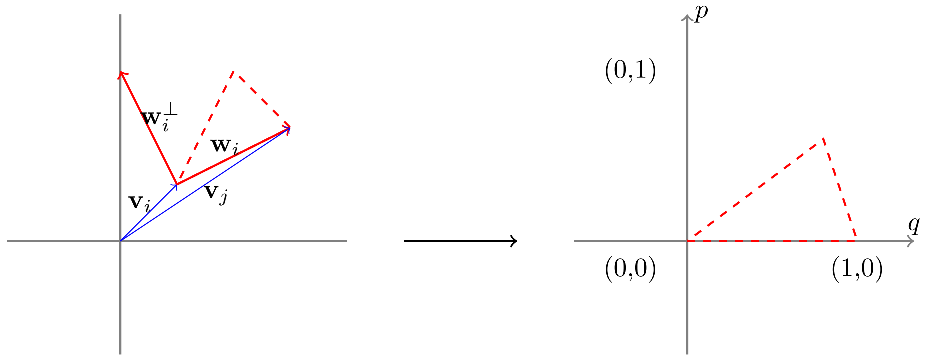


Figure 4.4: Reference-free parametrization.

step to describe U and V will be used exclusively in the remainder of the algorithm to calculate the intersections between the edges of U and those of V; which vertices of U lie inside V and; which vertices of V lie inside U. This idea of consistency is key for the robustness of PANG2.

4.3.1 Alternative to change of coordinates: Reference-free parametrizations

The change of coordinates described above is asymmetric over the vertices of V. The selection of a vertex of V as having position $\mathbf{v}_0$ is not unique, and depending on which of the three vertices are chosen three different geometries can result. However, this change of coordinates is not necessary. The parametrizations of the reference lines of Y may be found directly from the original coordinate system, without any transformation of V to Y.

To proceed, we first re-codify the positions of the vertices of V and U. Let the i–th vertex of V lie at the position defined by the vector $\mathbf{v}_i$, and let $\mathbf{u}_i$ represent the same for the i–th vertex of U. Let $\mathbf{w}_i$ be the vector running between the i–th and j–th vertices of V, where $j = i \pmod 3 + 1$.

As before, we seek an affine transformation that maps the i–th vertex to $(0,0)$ and the j–th vertex to $(1,0)$. Given that we are not concerned with the final position of the third vertex of V we ask only that the transformation avoids shearing to minimize possible error. Ultimately, the affine transformation for the i–th vertex of V satisfies

$$A\left(\mathbf{v}_i\mathbf{1}^\top + \begin{bmatrix}\mathbf{0} & \mathbf{w}_i & \mathbf{w}_i^\perp\end{bmatrix}\right) + \mathbf{b}\mathbf{1}^\top = \begin{bmatrix}0 & 1 & 0\\ 0 & 0 & 1\end{bmatrix},$$

where $\mathbf{w}_i^\perp$ is a vector orthogonal to $\mathbf{w}_i$. The transformation of $\mathbf{u}_k$, represented by (q_k, p_k), is found by solving

$$\begin{bmatrix}\mathbf{w}_i & \mathbf{w}_i^\perp\end{bmatrix}\begin{bmatrix}q_k\\ p_k\end{bmatrix} = \mathbf{u}_k - \mathbf{v}_i.$$

There is a degree of freedom in the choice of $\mathbf{w}_i^\perp$. It is natural to choose the triangle V to lie on the positive side ($p \geq 0$) of the line $p = 0$. To ensure this, we require that $\mathbf{w}_j^\top\mathbf{w}_i^\perp > 0$. This reduces to checking if the vertices of V are listed clockwise or counterclockwise.

An intersection $(q_0, 0)$ between this edge of V and a given edge of X must be transformed back into the original coordinates. This position is found using the formula $\mathbf{v}_i + q_0\mathbf{w}_i$.

FLOP count comparison

This reference-free parametrization of the edges of V is not recommended as it takes more floating-point operations (FLOPs) to run in practice. In this parametrization, each edge is treated equally, meaning no calculations from the other edges are used to speed up computing time. The change of coordinates described in Section 4.3 takes advantage of the degree of freedom used here to prevent shearing to combine two of the parametrizations and makes the third considerably cheaper to implement.

Under the reference-free parametrization, for each edge one must solve: a system of the form $A\hat{X} = B$ where A is a matrix of size 2×2 and $\hat{X}$ and B are of size 2×3; three checks of the sign bits of the p–coordinate of the vertices of X; up to two intersections between the line $p = 0$ and the edges of X; up to two transformations using the formula $\mathbf{v}_i + q_0\mathbf{w}_i$ and; two tests of the vertices of V. These last four steps will be explained in Section 4.4. Additionally, there is a check of the value of $\mathbf{w}_j^\top \mathbf{w}_i^\perp$, which constitutes 2 multiplications and 1 addition.

By comparison, using the change of coordinates only the last four of these steps need to be taken for each edge. This saves two system solves of the form $A\hat{X} = B$ and the check of $\mathbf{w}_j^\top \mathbf{w}_i^\perp$. However, the edge $x + y = 1$ needs special attention. As will be explained in Section 4.4 this edge has a pre-computed affine transformation of 4 additions and 1 multiplication for each vertex of X. Moreover, the transformations of the intersections along this edge use the form $\mathbf{v}_0 + (1 - q_0)\mathbf{v}_1 + q_0\mathbf{v}_2$. This represents a further 3 additions and 2 multiplications for each such intersection, of which there are at most two. The difference in computations therefore totals 2 fewer system solves, 17 more additions and 5 extra multiplications.

If the number of FLOPs to solve the system $A\hat{X} = B$ is more than 11, which is true of most methods to solve such systems, then the change of coordinates is a more efficient subroutine. This depends strongly on implementation.

Regardless of efficiency, the reference-free parametrization may prove more accurate. This parametrization presents two clear advantages to this point. Firstly, it is independent of the choice of vertex of V which is necessary for the change of coordinates. Secondly, the matrices used in the three affine transformations are shear-free.

4.4 Computation of the polygon of intersection

As explained in the introduction, the polygon of intersection for the triangles X and Y has three types of vertices: intersections between the edges of X and Y; vertices of X lying inside Y and; vertices of Y lying inside X.

The edge intersections of X and Y are the affine transformations of those of U and V and so may be reverse transformed to provide the latter. The locations of the vertices of U are known without reverse transformation, and so once the indices of which vertices of X lying inside Y are found they index also those of U that lie inside V. The same is true of the vertices of V lying inside U, using the indices of the vertices of Y lying inside X.

Parameters	$y = 0$	$x = 0$	$x + y = 1$
$p(x, y)$	y	x	$1 - x - y$
$q(x, y)$	x	y	$(1 - x + y)/2$

Table 4.2: Parameters $p(x, y)$ and $q(x, y)$ for each of the three reference lines of Y.

4.4.1 Edge Intersections

The reference triangle Y (the transformation of V) has two edges of length 1 aligned with the lines $x = 0$ and $y = 0$. The third edge runs along the line $x + y = 1$. These three lines will be called reference lines.

To standardize the calculations of the intersections and simplify the discussion, we construct parameters $p(x, y)$ and $q(x, y)$ for each reference line so that the reference line lies on $p = 0$, q increases orthogonally to p, and the edge of Y lies between $q = 0$ and $q = 1$. For the lines $x = 0$ and $y = 0$ this parametrization is trivial. Table 4.2 lists these parameters for each of the reference lines.

This transform from (x, y) to (q, p) represents another affine transformation. As opposed to the transformation in the previous section these transformations are known *a priori*. The most computationally intensive of these, for the line $x + y = 1$, is a rotation and translation. The condition number of a rotation is always 1 and so these affine transformations are stable.

We consider one edge of Y, its reference line, and the corresponding parameters $p(x, y)$ and $q(x, y)$. Let $p_i = p(x_i, y_i)$ and $q_i = q(x_i, y_i)$ be the values of the parameters for the i–th vertex of X. An intersection occurs between the reference line and the edge connecting the i–th vertex of X and its j–th vertex if p_i is positive and p_j is negative, or vice versa. We can then enforce the following condition: We will only calculate an intersection for the pair of vertices (i, j) if p_i and p_j have different signs.

The degenerate case where a vertex of X lies on the reference line needs to be considered. Without loss of generality, we suppose the first vertex of X lies on the reference line. By most conventions the sign of such a value of p_1 would be zero. If $p_2 > 0$ and $p_3 < 0$ then both pairs (1,2) and (1,3) have intersections with the reference line. As both of these are in fact the same point there is redundancy in these calculations. To avoid this, any vertex lying on the reference line will be considered to be on the positive side of the line. That is, we use the following binary-valued sign function:

$$\operatorname{sign}(p) = \begin{cases} 1 & p \geq 0, \\ 0 & p < 0, \end{cases} \tag{4.1}$$

as opposed to the ternary-valued sign function that allows a third value at 0. The condition $\operatorname{sign}(p_i) \neq \operatorname{sign}(p_j)$ is both necessary and sufficient for an intersection to exist, excepting the case where $p_i = p_j = 0$.

For the degenerate case where $p_i = p_j = 0$ there is an infinite number of intersections. However, to construct the polygon of intersection we need only the corners of the polygon. Therefore, there is no distinction between this degenerate case and the same configuration with p_i and p_j shifted an imperceptible distance away from the line in the positive p direction, as the polygon of intersection retains the same shape and size. Thus, the previous equivalence statement may ignore the case of an infinite number of intersections.

Suppose $\operatorname{sign}(p_i) \neq \operatorname{sign}(p_j)$. Then an intersection exists for the pair (i, j). To find the intersection it suffices to find the q–intercept of the line running through the two sets of parameters. There are a number of formulas to produce this result. For example, one may use the following:

$$q_0 = \frac{p_j q_i - p_i q_j}{p_j - p_i}. \tag{4.2}$$

This particular formula has the advantage of being symmetric in i and j. To test if the intersection is on the edge of the reference triangle, and not merely on the reference line, one tests if $q_0 \in [0, 1]$.

Proposition 4.1. *At most six intersections will be calculated under any triangle intersection algorithm that checks if* $\text{sign}(p_i) \neq \text{sign}(p_j)$. *The number of intersections calculated is even.*

Proof Consider the condition $\text{sign}(p_i) \neq \text{sign}(p_j)$. The function $\text{sign}(p)$ has only two possible values: 0 and 1. There are only two ways to partition three objects (p_1, p_2, p_3) into two sets (either 0 or 1): 3-0 and 2-1. No intersections are calculated for the first of these. For the other, there are two intersections. Thus, for a given reference line there is either 0 or 2 intersections calculated.

This proof may be applied to each of the three reference lines. If each condition produces the maximum two intersections, six intersections will be calculated. ■

Given the simple nature of the edges of the reference triangle Y the equations for the intersections are straightforward to write down:

$y = 0$**:** $(q_0, 0)$;

$x = 0$**:** $(0, q_0)$;

$x + y = 1$**:** $(1 - q_0, q_0)$.

This provides the transformation from (q, p) coordinates to (x, y) coordinates. To retrieve the coordinates of the intersections in the original coordinates, i.e. the coordinates of U and V, one multiplies the coordinate vector $\begin{bmatrix} x & y \end{bmatrix}^\top$ by the matrix $\begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 \end{bmatrix}$. For example, to retrieve the original coordinates of an intersection with the line $y = 0$, one performs the calculation

$$\begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 \end{bmatrix} \begin{bmatrix} q_0 \\ 0 \end{bmatrix}.$$

Note that the possible error of the position of the intersections with regard to the reference triangle are limited. The calculated intersection with the line $y = 0$ must lie on the line $y = 0$ and so there is no error in the vertical direction. The same is true of error in the horizontal direction for the intersection with the line $x = 0$ and of error along the line $x + y = 1$ for the intersection along this line.

4.4.2 Vertices of Y in X

To maintain consistency, we use the information from the floating-point calculations of the intersections to decide if the vertices of Y lie within X. First note that triangles are convex. Therefore it is both necessary and sufficient that a vertex of Y lies on a line between the boundaries of X for this vertex to lie inside of X.

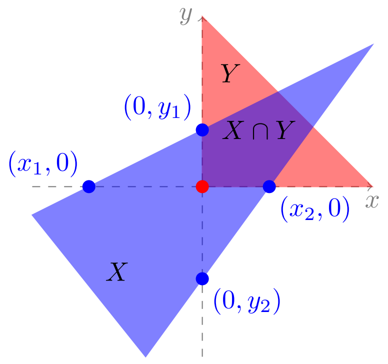


Figure 4.5: Intersections between the edges of X and the lines $y = 0$ and $x = 0$ surrounding the vertex of Y at the origin. For the line $y = 0$ the parameters q and p are x and y, respectively, and $q_0^1 = x_1$ and $q_0^2 = x_2$. For the line $x = 0$ $q = y$, $p = x$, $q_0^1 = y_1$ and $q_0^2 = y_2$.

By Proposition 4.1 there are at most two intersections for each line. Suppose a given line has two such intersections. One may differentiate between the two by using q_0^1 and q_0^2. Which pairs of i and j correspond to q_0^1 and q_0^2 is inconsequential.

Without loss of generality, suppose $q_0^1 < q_0^2$. The interval $[q_0^1, q_0^2]$ along the line $p = 0$ lies within the triangle X. Moreover, no points outside this interval and along this line lie within X. Two vertices of the triangle Y lie along this line, at $q = 0$ and $q = 1$. Therefore, a vertex of Y lies inside X if and only if $0 \in [q_0^1, q_0^2]$ or $1 \in [q_0^1, q_0^2]$. Figure 4.5 shows this process graphically.

If a given line has zero intersections, then it is impossible for a vertex of Y that lies along this line to be within X, as no part of this line lies within X. In this case the triangle X lies entirely on one side of the line or the other.

The tests of $q_0^1, q_0^2 \in [0, 1]$ and $0, 1 \in [q_0^1, q_0^2]$ simplify to checking whether $q_0 < 0$ and $1 - q_0 < 0$ for both q_0^1 and q_0^2. For intersections along the same edge of X there is correspondence between these tests.

Proposition 4.2. *The number of intersections lying on Y is even.*

Proof If all calculated intersections for a given reference line lie entirely inside or entirely outside $[0, 1]$ then Proposition 4.1 provides the result. However, if $q_0^1 \in [0, 1]$ and $q_0^2 \notin [0, 1]$ then we must prove that the same occurs for another reference line.

Without loss of generality, suppose $q_0^2 < 0$. Then $0 \in [q_0^2, q_0^1]$ and there must be two intersections on the second reference line running through the vertex at 0 such that they bound the same vertex, see Figure 4.5. Let these intersections be denoted $\hat{q}_0^1$ and $\hat{q}_0^2$, and let the vertex they bound be 1 such that $1 - \hat{q}_0^1 < 0$ and $1 - \hat{q}_0^2 \geq 0$. If $\hat{q}_0^2 \geq 0$ it forms a pair with q_0^2.

If instead $\hat{q}_0^2 < 0$ both vertices are bound by the intersections along this reference line. Thus, along the third reference line the second vertex is found between the two intersections, i.e. $1 - \tilde{q}_0^1 < 0$ and $1 - \tilde{q}_0^2 \geq 0$. Since $1 \notin [q_0^2, q_0^1]$ the third vertex is not found between the intersections of the third reference line and $\tilde{q}_0^2 \geq 0$. This intersection forms the necessary pair with q_0^2. ■

Consider the numerators of q_0 and $1 - q_0$ under the change of coordinates. These are presented in Table 4.3. Each is divided by their respective $p_j - p_i$. The values of the numerators are shared over calculations for the same edges of X. Thus, by ensuring the

Reference line	Numerator of q_0	Numerator of $1-q_0$
$y=0$	$x_iy_j - x_jy_i$	$(1-x_i)y_j - (1-x_j)y_i$
$x=0$	$y_ix_j - y_jx_i$	$(1-y_i)x_j - (1-y_j)x_i$
$x+y=1$	$(1-x_j)y_i - (1-x_i)y_j$	$(1-y_j)x_i - (1-y_i)x_j$

Table 4.3: Numerators of q_0 and $1-q_0$ for the three reference lines of Y. These have been simplified and use the change of coordinates.

calculations agree on the signs of these numerators the algorithm can make a consistent determination on which side of a vertex of Y the edge of X falls.

While the value of q_0 is used to calculate the intersections, we need only the sign of $1-q_0$ and compare it with the sign of p_j. Each of these numerators, of which there are nine in total, can be computed as a determinant. The numerators for the entire triangle X can be computed as cross products.

Under the reference-free parametrization the correspondence is less obvious but is still present. Let $\mathbf{w}_i$, $\mathbf{v}_i$, $\mathbf{u}_k$ and j be as defined in Section 4.3.1, and let q_k^i and p_k^i be the transformation of $\mathbf{u}_k$ for the i–th vertex of V. Then

$$\begin{aligned}
\begin{bmatrix}\mathbf{w}_i & \mathbf{w}_i^{\perp}\end{bmatrix}\begin{bmatrix}1-q_k^i \\ p_k^i\end{bmatrix} &= \mathbf{w}_i + \begin{bmatrix}\mathbf{w}_i & \mathbf{w}_i^{\perp}\end{bmatrix}\begin{bmatrix}-1 & 0 \\ 0 & 1\end{bmatrix}\begin{bmatrix}q_k^i \\ p_k^i\end{bmatrix} \\
&= \mathbf{w}_i + \begin{bmatrix}\mathbf{w}_i & \mathbf{w}_i^{\perp}\end{bmatrix}\begin{bmatrix}-1 & 0 \\ 0 & 1\end{bmatrix}\begin{bmatrix}\mathbf{w}_i & \mathbf{w}_i^{\perp}\end{bmatrix}^{-1}(\mathbf{u}_k - \mathbf{v}_i) \\
&= \mathbf{w}_i + \begin{bmatrix}\mathbf{w}_i & \mathbf{w}_i^{\perp}\end{bmatrix}\begin{bmatrix}-1 & 0 \\ 0 & 1\end{bmatrix}\begin{bmatrix}\mathbf{w}_i & \mathbf{w}_i^{\perp}\end{bmatrix}^{-1}(\mathbf{u}_k - \mathbf{v}_j + \mathbf{w}_i) \\
&= \begin{bmatrix}\mathbf{w}_i & \mathbf{w}_i^{\perp}\end{bmatrix}\begin{bmatrix}-1 & 0 \\ 0 & 1\end{bmatrix}\begin{bmatrix}\mathbf{w}_i & \mathbf{w}_i^{\perp}\end{bmatrix}^{-1}\begin{bmatrix}\mathbf{w}_j & \mathbf{w}_j^{\perp}\end{bmatrix}\begin{bmatrix}q_k^j \\ p_k^j\end{bmatrix}.
\end{aligned}$$

Note that the choice of $\mathbf{w}_j^{\top}\mathbf{w}_i^{\perp} > 0$ made in Section 4.3.1 forces the determinant of the product of $\begin{bmatrix}\mathbf{w}_i & \mathbf{w}_i^{\perp}\end{bmatrix}^{-1}$ and $\begin{bmatrix}\mathbf{w}_j & \mathbf{w}_j^{\perp}\end{bmatrix}$ to be positive, see Lemma 4.11 in Appendix 4.13. Therefore, for a given edge of X the numerator of $1-q_0$ for the i–th vertex of V has the opposite sign of the numerator of q_0 for the j–th vertex of V. Both numerators measure the distance from the j–th vertex of V.

As a protective measure, we test whether the edge of X intersects neighbouring reference lines. Let p_i' and p_j' be the p–coordinates of the i–th and j–th vertices of X, respectively, for the neighbouring reference line. If the edge between these vertices does not intersect the neighbouring reference line then $\operatorname{sign}(p_i') = \operatorname{sign}(p_j')$. Moreover, they are both equal to $\operatorname{sign}(q_0)$ or $\operatorname{sign}(1-q_0)$, depending on the line, see Figure 4.6. By performing this check we prevent intersection errors involving edges of X that intersect only one reference line of Y, see Sections 4.5.1 and 4.6.2.

4.4.3 Vertices of X in Y

Vertices of X in Y cannot be found using the same approach as the two problems are not symmetric: We do not have the same information from the intersection calculations. However, there is sufficient information to make consistent determinations using a different approach.

The parameters $p(x,y)$ have been set up such that the reference triangle Y lies on the positive side of the line $p=0$. That is, if $(x,y) \in Y$ then $p(x,y) \geq 0$ for each of

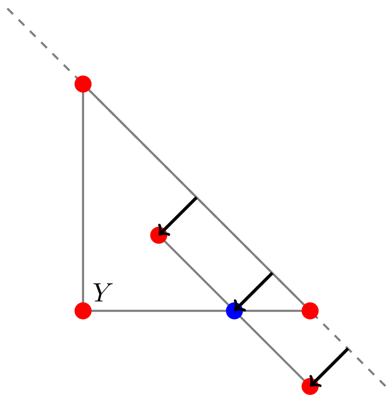


Figure 4.6: Test of sign of $1-q_0$ without computation. In this example, both vertices of the edge of X lie on the positive side of the reference line $x+y=1$. The intersection must also, implying $1-q_0>0$.

the three reference lines. Conversely, if $p(x,y) \geq 0$ for all of the reference lines then $(x,y) \in Y$. Therefore, the i–th vertex of X lies inside Y if and only if $\text{sign}(p_i)=1$ for all three reference lines. One can keep track of this at each edge by creating a dummy variable, s, originally set to 1 and multiplied by $\text{sign}(p_i)$ during each edge parametrization. Since $\text{sign}(p_i)=0$ if $p_i<0$ the final result of s will be 0 if any $p_i<0$ and will be 1 if and only if $p_i \geq 0$ for every parameter $p(x,y)$.

Because the values of p_i are also used in the floating-point calculations of the intersections, this step is consistent with these calculations and, by extension, the determination of the vertices of Y in X. The three types of points in the polygon of intersection make use of the same information, and a numerical error in part of this information affects each type in a consistent way.

4.5 Robust algorithm for 2D triangle intersections

We will now write the intersection algorithm of PANG2 in full.

Algorithm 4.1 (PANG2). **Step 1: Change of coordinates.** *An affine transformation is used to change the coordinates of the vertices of X into a reference frame whereby Y is the reference triangle described in Section 4.3.*

Step 2: Select reference line. *Choose a reference line of the reference triangle. Apply the correct functions for the parameters and make note of which vertices of Y lie on this line.*

2(i): Intersections. *Test if* $\text{sign}(p_i) \neq \text{sign}(p_j)$. *If so, calculate the intersection with the reference line and test $q_0<0$ and $1-q_0<0$. If both are false, the intersection lies on the edge of Y. Repeat this step for all three pairs of vertices of X. By Proposition 4.1 this results in at most two unique intersections. One may remove duplicates at this stage but it is not necessary.*

2(ii): Vertices of Y in X. *Use the tests of $q_0<0$ to determine if $0 \in [q_0^1, q_0^2]$. If it does, the corresponding vertex of Y lies within X. Repeat for the vertex at $q=1$.*

Repeat step 2 for each of the three reference lines.

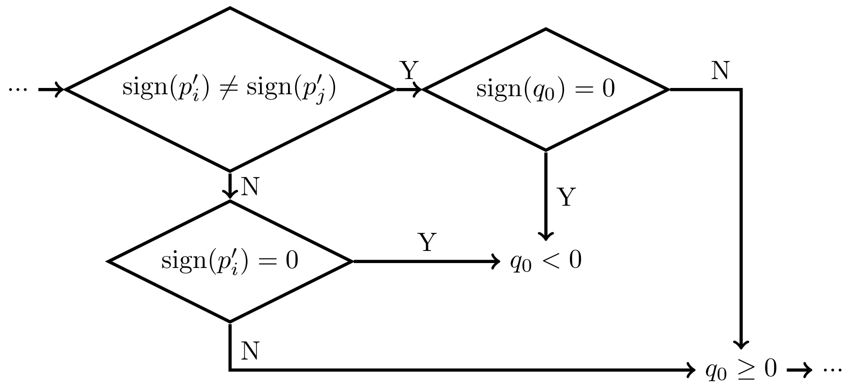


Figure 4.7: Decision tree for the test of $q_0 < 0$. The test of $1 - q_0 < 0$ is identical.

Step 3: Vertices of X in Y. *Multiply the three values of* $\text{sign}(p_i)$ *together for each of the three vertices of X. The result will either be 0 or 1. If it is 1, the vertex lies inside Y.*

Step 4: Reverse change of coordinates. *The vertices of X and Y in the standard coordinates are already known and so one can take those vertices determined to be in Y and X, respectively, without additional calculations. For the intersections one must apply the inverse affine transformation of the first step of the algorithm.*

4.5.1 Test of $q_0 < 0$

We outline the implementation of step 2(i) of the algorithm, where we test if $q_0 < 0$ and $1 - q_0 < 0$. Without loss of generality we consider only the test of $q_0 < 0$. As explained in Section 4.4.2 if the signs of p'_i and p'_j, the p–coordinates of the vertices of X for the neighbouring reference line, are equal then $\text{sign}(q_0) = \text{sign}(p'_i) = \text{sign}(p'_j)$. The first test is then to check whether $\text{sign}(p'_i) = \text{sign}(p'_j)$. Depending on this result we either use the value of $\text{sign}(p'_i)$ to determine if $q_0 < 0$ or we test directly $\text{sign}(q_0)$.

Figure 4.7 shows the decision tree of the test of $q_0 < 0$. As outlined above, there are three determinations to be made: whether $\text{sign}(p'_i) \neq \text{sign}(p'_j)$; $\text{sign}(q_0)$ and; $\text{sign}(p'_i)$. To make the code compact these three decisions can be combined into a single logical test:

$$[\text{sign}(p'_i) \neq \text{sign}(p'_j) \wedge \text{sign}(q_0) = 0] \vee [\text{sign}(p'_i) = 0 \wedge \text{sign}(p'_j) = 0] \implies q_0 < 0.$$

If this test returns that $q_0 < 0$ then this information must be used to determine if the vertex of Y at $q = 0$ lies in X. We store this information and await the result of the test of the other intersection on this reference line. If this second test returns $q_0 \geq 0$ then the vertex of Y lies in X.

4.6 Consistency errors

The triangle intersection algorithm presented in Section 4.4 calculates the polygon of intersection for a right angled reference triangle Y and an arbitrary triangle X using a single consistent geometry. Combined with the change of coordinates of Section 4.3 it may be used to find the polygon of intersection for any two triangles U and V, again with a consistent geometry for all calculations. It remains to show that any numerical errors incurred throughout the calculations maintain the consistency of the algorithm, namely that the algorithm produces an intersection of similar size and shape to the one that exists between U and V.

There are three errors related to consistency that can occur in this algorithm. Firstly, a vertex of X may be found to lie inside (resp. outside) of Y when it is meant to lie outside (inside) (X-in-Y error). Secondly, an intersection of an edge of X with a reference line of Y may be found to lie on (off) the edge of Y when it is meant to lie off (on) (intersection error). Thirdly, a vertex of Y may be found to lie inside (outside) of X when it is meant to lie outside (inside) (Y-in-X error). We examine each of the three errors independently, while noting where applicable when one error can cause another.

4.6.1 X-in-Y errors

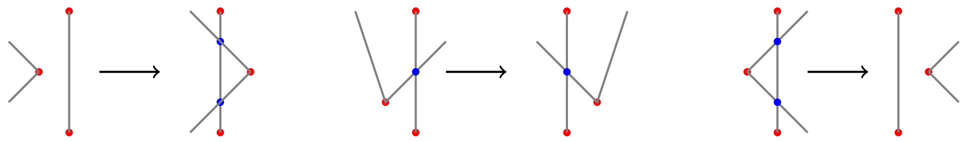
It is important that when an X-in-Y error occurs an appropriate number of intersections is calculated. Whether the intersections are deleted or created, there should be the correct number associated with the intersection of two triangles, albeit slightly altered. The positions of the intersections may be in error, but this will be considered later as a separate error. The following lemma provides that if the algorithm suffers an X-in-Y error then there will be a corresponding change in the number of intersections calculated.

Lemma 4.3. *Suppose the i–th vertex of X lies outside the reference triangle Y but is determined to be within Y by the algorithm described in Section 4.5. There is at least one reference line between the correct and calculated positions of the i–th vertex. An intersection is calculated by the algorithm between this reference line and the edge of X between the i–th and j–th vertices if and only if this reference line and edge do not intersect.*

Proof There are either one or two reference lines between the correct and calculated positions of the i–th vertex of X. One of these reference lines has parameters $p(x, y)$ and $q(x, y)$, see Table 4.2. The correct value of $\text{sign}(p_i)$ is 0 but the algorithm has returned $\text{sign}(p_i) = 1$. The value of $\text{sign}(p_j)$ must equal one of these. If $\text{sign}(p_j) = 1$ then the edge of X intersects the reference line but no intersection is calculated. If $\text{sign}(p_j) = 0$ then the edge does not intersect the reference line but an intersection is calculated. ■

Lemma 4.3 remains true if the correct and calculated positions of the i–th vertex are reversed. This lemma does not give an indication as to whether these intersections lie on the edges of Y. Geometrically, such an intersection must lie on an edge of Y. However, the algorithm allows the possibility of error. Such intersection errors are considered separately in Section 4.6.2.

For a vertex of X that suffers an X-in-Y error there are two other vertices of X. The effects of Lemma 4.3 therefore occur twice for each error of this type. Figure 4.8 shows the possible combinations of two applications of Lemma 4.3 when the intersections occur over the same edge of Y. In each of these figures, the triangle vertex of X moves to the right to cross the reference line of Y. As it does so, two intersections are created and move further apart (Figure 4.8a); the intersection slides along the edges



(a) No edges intersect to both edges intersect. (b) One edge intersects to the other edge intersects. (c) Both edges intersect to no edges intersect.

Figure 4.8: Results of X-in-Y errors.

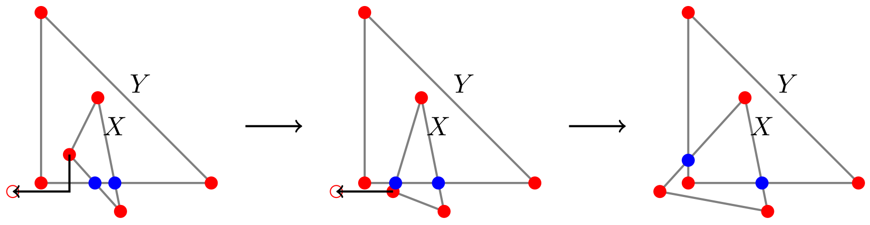


Figure 4.9: The cascade of errors used to describe when an X-in-Y error alters intersections on two separate reference lines. An intermediate position between the correct and calculated positions of the i–th vertex is presented in the centre of the figure.

of the triangles, crossing over the vertex (Figure 4.8b) or; the two intersections move towards each other until they meet at the moment the vertex crosses the reference line, disappearing (Figure 4.8c).

If the intersections occur over separate edges of Y then we must consider the result to be a cascade of errors. We consider an intermediate position of the i–th vertex of X such that the movement between the correct and intermediate positions creates intersections along a single edge of Y and the movement between the intermediate and calculated positions causes two intersection errors that result in a Y-in-X error, see Figure 4.9.

4.6.2 Intersection errors and Y-in-X errors

A small change in the position of an intersection can only cause an error in consistency if it passes over a vertex of Y. In such a case, the intersection either enters or leaves the polygon of intersection. We call this an intersection error. Since this algorithm uses the intersections to determine which vertices of Y lie in X, this can cause Y-in-X errors.

We begin examining intersection errors by dividing them into two categories, paired and unpaired. A paired intersection error occurs when two intersections along the same edge of X both suffer intersection errors over the same vertex of Y. An unpaired intersection error is a single intersection error that occurs on its own. We prove that paired intersection errors preserve consistency and that unpaired intersection errors are impossible.

There are 6 configurations of paired intersection errors, see Figure 4.10. An intersection error represents a reversal in sign in either q_0 or $1 - q_0$ for a given edge of X. Without loss of generality, we suppose it is in q_0. Recall from Table 4.3 that the numerator of q_0 is shared with that of another intersection of the same edge of X. Thus, if the sign of q_0 is in error then the sign of this second intersection along this edge of X is also reversed. This pairs the intersection errors.

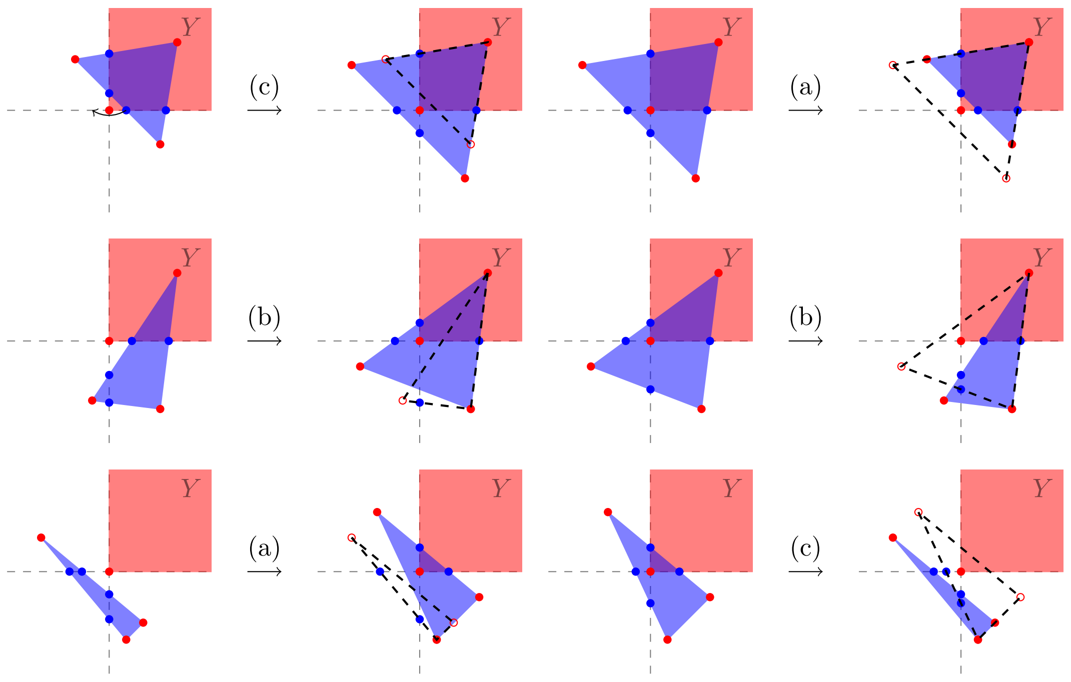


Figure 4.10: Paired intersection errors. These errors have the same results as X-in-Y errors. The corresponding results of Figure 4.8 are referenced for each configuration.

An unpaired intersection error over a vertex of Y can only occur if the edge of X does not intersect the other reference line that extends from the vertex. Then, in the parametrization for this other reference line, the p–coordinates of the two vertices of X attached to this edge are either all positive or all negative. The algorithm tests for this to prevent the error, see Section 4.5.1. In this way, unpaired intersection errors cannot occur.

Figure 4.10 indicates that even under intersection errors a vertex of Y lies within X if and only if it is surrounded on all sides by four intersections. Tests of Y vertices along neighbouring reference lines return consistent results. Proposition 4.2 is therefore unaffected by errors.

Lemma 4.4. *Let q_0 be the position of the intersection between the i-th and j-th vertices of X along a given reference line of Y. Without loss of generality, let $p_j \geq 0$ and $p_i < 0$. Suppose the numerator of q_0 has floating-point error δ from its calculation due to equation (4.2). Then the error in the area of the polygon of intersection in the transformed coordinates due to equation (4.2) is $\mathcal{O}(\delta)$.*

Proof Since the error in the numerator of q_0 is δ, the error in q_0 is $\delta/(p_j - p_i)$. Given that $\text{sign}(p_j) \neq \text{sign}(p_i)$ there is no cancellation error in $p_j - p_i$ and the error in the denominator is negligible.

The coordinates $(q_0, 0)$, $(q_0 + \delta/(p_j - p_i), 0)$ and (q_j, p_j) form a triangle, denoted X_δ. This triangle has a base of $\delta/(p_j - p_i)$ and a height of at most p_j and so has an area of at most $\delta p_j / 2(p_j - p_i) < \delta/2$. It suffices then to show that this triangle represents the largest possible change in the polygon.

Consider the polygon of intersection before the effects of the error. In particular, consider the two neighbours of the intersection $(q_0, 0)$. One of these neighbours lies along the same reference line and is either another intersection or a vertex of Y. The

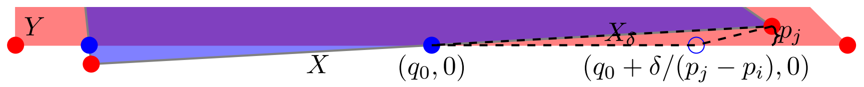


Figure 4.11: Error of an intersection amplified in equation (4.2) is mitigated by the position of a neighbouring node.

other lies along the edge of the triangle X between its i-th and j-th vertices. Therefore, it is either the j-th vertex of X or an intersection between this edge of X and another reference line. Such an intersection would have a height less than p_j.

After the effects of error, the intersection has shifted. One of the edges of the polygon that extend from this intersection has shrunk while the other has grown. We seek a triangulation of the polygon such that its change is represented by a single triangle. Since one of the neighbouring corners of the polygon lies along the reference line containing both $(q_0, 0)$ and $(q_0 + \delta/(p_j - p_i), 0)$ this single triangle is defined by these nodes and the other neighbour. As previously stated this other neighbour lies on the same edge of X as $(q_0, 0)$ and the j-th vertex of X. The triangle representing the change in the polygon then lies within the triangle X_δ and so has an area of at most $\mathcal{O}(\delta)$. ■

While $p_j - p_i$ is not subject to cancellation error, the numerator of q_0 is. The magnitude of δ is then $\mathcal{O}(L^2\epsilon_m)$, where L is the longest edge of X and ϵ_m is machine epsilon. The value of L is a measure of the relative sizes of the original triangles U and V. If the two triangles are of roughly equal diameter then this magnitude is 1.

4.7 Graphs of triangle-triangle intersections in 2D

Section 4.6 indicates that each error of the algorithm can cause a change in the shape of the polygon of intersection. It must be shown that these changes in shape can cause no significant changes to the area of the intersection. That is, a change in the shape of the polygon must be in some way continuous.

Our strategy to prove this is to construct the set of graphs that represent the possible intersections of two triangles and show that the errors transform these graphs between each other and not outside the set. In this way we will show that at worst the calculated intersection corresponds to two triangles that are of slightly different size, shape and position.

We first note that PANG2 is valid: Given a pair of triangles and assuming no numerical error in any of the calculations the algorithm will correctly identify the intersection between them. Therefore, we need only consider the set of possible intersections and how the errors affect this set. Thus, the first step is to determine this set.

While the intersection of two triangles is a purely geometric concept, the calculation of such an intersection is subject to numerical error. This error can cause vertices and intersections to move. The resulting polygon of intersection may not lie precisely within the edges of the triangles. We therefore construct graphs of the triangle intersections. In this way the intersections need not be geometric, allowing for the possibility of error.

As these graphs represent geometric objects in the 2D plane the graph edges are straight. We distinguish between two types of graphs: those of geometric intersections and those of intersections calculated by the algorithm. Hoffmann [26] refers to the first as models and the second as representations. The algorithm is robust if the models are close to the representations. Ideally, the set of models should be coincident with the set of representations.

Both types of graphs satisfy a number of conditions.

Vertex condition: triangle vertices (of which there are exactly 6 and are coloured red) are nodes with two neighbours.

Intersection condition: edge intersections (which may number 0, 2, 4 or 6, see Proposition 4.2, and are coloured blue) are nodes with four neighbours.

Triangle condition: the triangle vertices are divided into two disjoint 3-cycles, each of which corresponds to one of the triangles. The points of overlap of these cycles are the edge intersections.

The triangle condition is so named because it ensures that the graph represents the intersection of triangles rather than some other pair of shapes.

In addition, when degenerate cases are removed (as may be done with this algorithm, to be discussed in Section 4.7.1) the models possess the following property.

Sheltered polygon property: no graph edge between two intersection nodes touches the exterior of the graph. That is, in the dual graph there is no graph edge between the node representing the polygon of intersection and the node representing the exterior of the graph.

The only way for the sheltered polygon property to be violated is for two edges of the triangles to be coincident, a degenerate case considered in the previous sections. Representations also satisfy the sheltered polygon property.

Lemma 4.5. *Any representation of this algorithm has the sheltered polygon property.*

Proof By construction, the nodes of the polygon of intersection as calculated by the algorithm lie interior to or on the boundary of Y. As such, a graph edge between two intersection nodes is either interior to Y or lies on a reference line of Y. In the former case, the interior of Y shelters the graph edge from the exterior.

In the latter case, these intersection nodes were calculated because two edges of X intersect this reference line, such that $\text{sign}(p_i) = \text{sign}(p_j) \neq \text{sign}(p_k)$. Without loss of generality, suppose $\text{sign}(p_k) = 0$ and the k–th vertex of X lies outside the triangle Y. Then there is a graph edge between this vertex and each of the intersection nodes. This shelters the graph edge between the intersection nodes from the exterior of the graph. ■

This ignores a number of degenerate cases where edge intersections and triangle vertices overlap. It will be shown in Section 4.7.1 that all degenerate cases may be represented by non-degenerate cases, all of which satisfy these three conditions and the sheltered polygon property. Figure 4.12 gives graphical examples of the triangle condition and the sheltered polygon property. It also illustrates the proof of Lemma 4.5. The graph has one graph edge between intersection nodes. If the blue triangle is Y then this graph edge is interior to Y and therefore sheltered from the exterior. If the blue triangle is X then the portion of X that does not intersect Y shelters the graph edge from the exterior. Numerically this portion of X must be there or the intersections would not be calculated.

Figure 4.13 shows all graphs representing two intersecting triangles. Not shown is the graph of two non-intersecting triangles, which is two disjoint graphs of 3-cycles of red vertices.

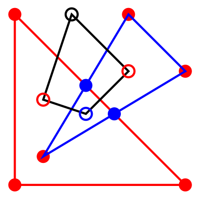


Figure 4.12: Graphic representation of the triangle condition and sheltered polygon property for a given graph. The two cycles of the triangle condition are shown with red and blue edges, and the dual graph is shown with black edges.

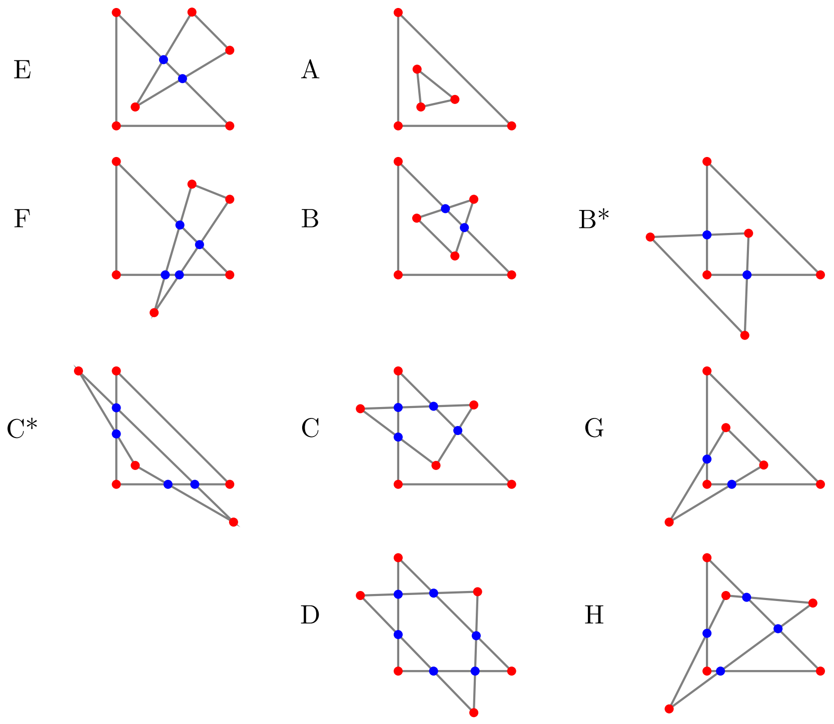


Figure 4.13: The models arising from all possible triangle-triangle intersections, degenerate cases removed.

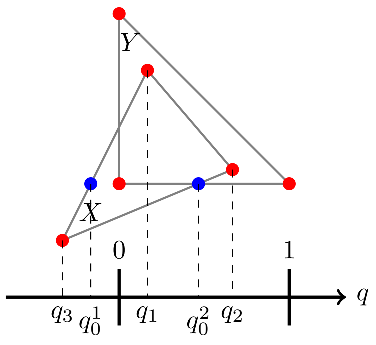


Figure 4.14: If two vertices of X lie inside Y then at most one vertex of Y can lie inside X.

Lemma 4.6. *Figure 4.13 is an exhaustive list of all possible models.*

Proof We begin by proving a restriction on the number of triangle vertices in the polygon of intersection: There can be at most three triangle vertices in the polygon, split into at most two groups, e.g. A and G.

Any collection of X vertices in the triangle Y necessarily connect to one another without any intersections between them, as there are no reference lines of Y inside of Y. As such, in the graph of the polygon of intersection, they are adjacent to one another. The same is true of Y vertices in the triangle X. Therefore, triangle vertices are divided into at most two groups.

Suppose two vertices of X lie inside Y. Without loss of generality, Y is the reference triangle. Choose a reference line of Y, with parameters p and q. The q–coordinates of the vertices of X that lie inside Y are necessarily between 0 and 1. Let q_0^1 and q_0^2 be the q–coordinates of the edges of X with this reference line. If q_0^1 is less than 0 then q_0^2 is less than 1, see Figure 4.14. Therefore, only one vertex of Y can lie inside X. This is symmetric: If there are two vertices of Y inside X then at most one vertex of X can lie inside Y.

Next, we prove that the number of intersections in the polygon is less than or equal to the number of triangle vertices outside of the polygon. Consider an intersection in the polygon and suppose one traverses the two cycles we know to exist by the triangle condition in a clockwise direction. At this starting intersection one cycle exits the polygon. At the next intersection the cycle enters the polygon. For the cycle to do so it must meet a corner between these intersections. Thus, there must be at least one vertex between the intersection of exit and the intersection of entry. Since each intersection is an intersection of exit for one of the two cycles, each intersection represents at least one triangle vertex outside of the polygon.

The vertex and intersection conditions and the two previous results may be combined in the following list of restrictions on the number of intersections n and the number of triangle vertices m in the polygon.

i. $n \in \{0, 2, 4, 6\}$ (intersection condition);

ii. $m \leq 3$ (previous result);

iii. $3 \leq n + m$ (polygons have at least three vertices);

iv. $n + m \leq 6$ (triangle condition and previous result).

m	n			
	0	2	4	6
3	A	G, G†	▨	▨
2	▨	B, B*	H, H†, H†*	▨
1	▨	E	C, C*	▨
0	▨	▨	F	D

Table 4.4: Graphs corresponding to each n-m pairing. The graphs indicated by the † symbol do not satisfy the necessary conditions and are found in Figure 4.23 in Appendix 4.11. The remainder are found in Figure 4.13.

We can then ascertain that only the following n-m are permitted: 0-3, 2-1, 2-2, 2-3, 4-0, 4-1, 4-2, and 6-0.

To determine which, if any, graphs correspond to these combinations, we present the following algorithm to construct a triangle–triangle intersection graph from a polygon of intersection.

Algorithm 4.2. *1. Draw the polygon of intersection using one of the eight combinations provided. Only three combinations provide multiple possible polygons: 2-2, 2-3 and 4-2. One of the options for the combination 2-3 may be immediately discarded as it contains three triangle vertices in a row, which is an entire triangle and thus does not have any intersections, see Figure 4.23a in Appendix 4.11.*

2. *Connect each adjacent pair of intersections by a graph edge exterior to the polygon. A pair of intersections is adjacent if they lie next to each other or if they are separated only by triangle vertices.*

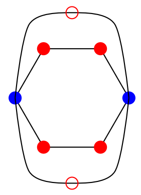
3. *Following the cycles defined by the triangle condition, add either one or two triangle vertices to each of the graph edges created by the previous step such that each cycle contains the necessary three triangle vertices. This will exclude two of the 4-2 combinations, as some cycles will have at least four triangle vertices, see Figures 4.23b and 4.23c in Appendix 4.11. Only one combination, 4-1, has two possible ways of satisfying all conditions of these graphs.*

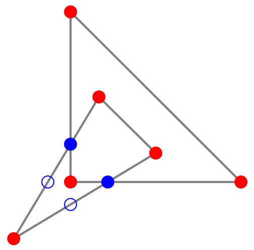
In this way the eight permitted combinations result in the ten graphs of Figure 4.13. Table 4.4 identifies each n-m pairing with either one of the graphs of Figure 4.13 or one of the invalid configurations of Figure 4.23 found in Appendix 4.11. ■

Lemma 4.6 does not immediately apply to intersections calculated by the algorithm because it relies on geometric arguments. In particular, the argument used to fix restriction (ii) $m \leq 3$ only applies when points of intersection lie strictly on the edges of the triangles. Numerical error can warp the positions of intersections to include or exclude triangle vertices in the polygon of intersection.

Lemma 4.7. *Restriction (ii) applies to representations.*

Proof An intersection node in the polygon of intersection lies between two vertices of the triangle Y. The pair of intersections along the reference line that passes through these two vertices then does not enclose at least one of these vertices. If there are two intersections on the reference line neither vertex is enclosed. Each pair of intersections then excludes at least one vertex of Y from the polygon of intersection.


(a) I†, the only possible 2-4 graph.


(b) The graph G.

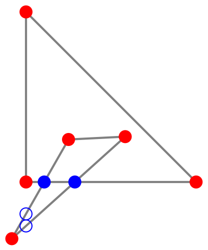
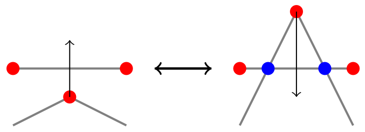

(c) The graph B.

Figure 4.15: The 2-4 pairing is shown to be invalid for representations.


(a) First type of graph rewrite.

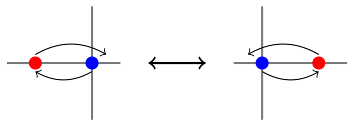

(b) Second type of graph rewrite.

Figure 4.16: The two types of graph rewrites codified by the algorithm.

Therefore, if there are two vertices of Y in the polygon then there are at most two intersection nodes. Since the only way to violate restriction (ii) is to have $m = 4$ with two vertices of Y and two of X, the only new n-m to consider is 2-4.

By the same argument that eliminated graph G† there is only one possible graph with a 2-4 pairing, I†, see Figure 4.15a. This graph does not violate any of the required conditions. Two of the triangle vertices in the polygon of intersection are necessarily those of X, while the other two are those of Y. There are only two graphs of Figure 4.13 with exactly two vertices of X inside Y: graphs B and G.

In graph G any intersection error is paired. Refer to the middle row of Figure 4.10 for the results of one or two of these errors. Under one paired intersection error the graph G becomes the graph B, see Figure 4.15c. Under a second error the graph B reverts to the graph G, see Figure 4.15b.

In graph B either the two intersections are both the only intersections along their respective edges of X or the edges of X both intersect another reference line. In the first case, any intersection errors are unpaired and thus impossible. In the second case, a paired intersection error results in graph G, see Figures 4.15b and 4.15c. A second error results in a different graph B, with the intersections on the other reference line. ■

Corollary 4.8. *Figure 4.13 is an exhaustive list of all possible representations.*

Proof It suffices to prove the four restrictions of the proof of Lemma 4.6 apply to representations. Restriction (i) applies by the intersection condition. By Lemma 4.7 restriction (ii) applies. Restriction (iii) applies naturally. The sheltered polygon property ensures the reasoning behind restriction (iv) applies. ■

Figure 4.16 shows the two graph rewrites codified by errors in the algorithm. Each rewrite possesses an inverse, as indicated by the double headed arrows in the figure. Figure 4.16a corresponds to Figures 4.8a and 4.8c, while Figure 4.16b corresponds to Figure 4.8b. Recall these figures detail the results of X-in-Y errors and intersection

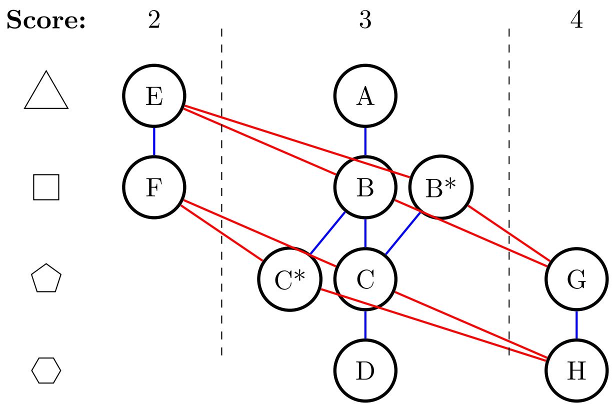
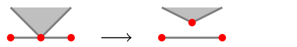
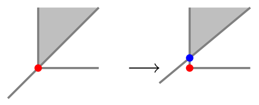


Figure 4.17: Graph of intersection connectivity: A blue edge between two graphs indicates one may be achieved through a first type rewrite of the other, while a red edge indicates a second type rewrite is required.


(a) Vertex of X on an edge of Y.


(b) Vertex of Y on an edge of X.

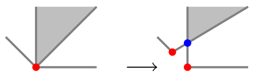

(c) Vertices of X and Y coincident.

Figure 4.18: Degenerate cases. The shaded area represents the intersection between X and Y.

errors. The correspondence between the intersection errors and their respective rewrites is indicated in Figure 4.10.

Figure 4.17 shows how the graphs of Figure 4.13 are connected through the rewrites of Figure 4.16. The graphs are organized in two ways. Each row has the polygon of intersection increasing in number of edges: graphs E and A have triangular polygons, graphs F, B and B* have quadrilateral polygons, and so on.

The columns are more complicated. One may assign point values to the vertices of the polygon of intersection: one point for each triangle vertex and half a point for each edge intersection. The columns of the graph of Figure 4.17 then have increasing point value sums or 'scores', with graph E having two points and graph H having four points.

The organization of this graph indicates important features of the two graph rewrites: Both rewrites add or subtract a graph edge to or from the polygon of intersection; and the second type rewrite changes the score of the polygon, adding or subtracting one. A score increase is coincident with a graph edge addition, so that when the score increases so does the number of graph edges of the polygon. The polygon of intersection therefore determines the position of the graphs of Figure 4.13 in the graph of Figure 4.17.

4.7.1 Degenerate cases

There are three major degenerate cases to consider: a vertex of X lying on an edge of Y; a vertex of Y lying on an edge of X and; a vertex of X coincident with a vertex of Y.

Should a vertex of X lie on an edge of Y it is considered to be inside of Y by the algorithm. An intersection is only calculated if another vertex lies on the negative side of this edge. For such an intersection the algorithm will calculate it as the vertex. When the copies of the vertex are removed, we will be left with a single point, a vertex of X in Y. Thus, this degeneracy reduces to the vertex of X being just inside the triangle Y. If the calculated intersection does not align with the vertex then the degeneracy is completely removed as the algorithm has created a numerical distance between the vertex and the edge by introducing an intersection between them.

If a vertex of Y lies on an edge of X, the algorithm considers an edge intersection to be coincident with the vertex of Y. This is ultimately equivalent to the same configuration with the edge intersection moved slightly along one of the two reference lines extending from the vertex of Y, such that the vertex is not considered to be inside of X. This keeps the same number of edges of the polygon of intersection and the same overall shape of the two triangles.

If a vertex of X is coincident with a vertex of Y we can consider both degenerate cases described above to have occurred. The degeneracy can be considered as a standard configuration with the vertex of X lying just outside of Y and the edge intersection moved in the same manner as when a vertex of Y lies on an edge of X.

4.8 Main results

We assume that we encounter neither overflow nor underflow.

Theorem 4.9. *Let $f(U,V)$ be the area of the intersection between triangles U and V calculated by the algorithm described in Section 4.5, where V is transformed into a reference triangle. Let L be the ratio of the diameters of U and V, α the smallest angle in V, $\alpha \le \pi/3$, and C the ratio of the lengths of the sides of V that form this angle, $C \le 1$. Let ΔU and ΔV be perturbations in the positions of the vertices of U and V, respectively. If the relative change in V is smaller than $C\sin(\alpha/2)$ then*

$$|f(U+\Delta U, V+\Delta V) - f(U,V)| = \frac{1+L}{C\sin(\alpha/2) - \mathcal{O}\left(\Delta V/V\right)}\mathcal{O}\left(\Delta U + L\Delta V\right) + \mathcal{O}\left(\Delta V\right).$$

Recall that the matrix A is the inverse of $\begin{bmatrix}\mathbf{v}_1 & \mathbf{v}_2\end{bmatrix}$. The vectors $\mathbf{v}_1$ and $\mathbf{v}_2$ run along the edges of V, see Section 4.3. The condition number of the matrix A is less than $1/(C\sin(\alpha/2))$ where α is the angle between $\mathbf{v}_1$ and $\mathbf{v}_2$ and C is the ratio between their lengths, $C \le 1$ [26]. Theorem 4.9 may then be restated in terms of the matrix A. To simplify notation, let $\hat{U}$, $\hat{V}$ and $\hat{X}$ denote the matrices containing the positions of the vertices of the triangles U, V and X, respectively.

Theorem 4.10. *If $\|A\| \left\|\Delta\hat{V}\right\| < 1$ then*

$$|f(U+\Delta U, V+\Delta V) - f(U,V)| = \frac{\kappa(A)\,(1+L)}{1-\kappa(A)\mathcal{O}\left(\Delta V/V\right)}\mathcal{O}\left(L\Delta V + \Delta U\right) + \mathcal{O}\left(\Delta V\right).$$

Proof As stated in Section 4.3 the positions of the vertices of X are determined by solving the linear system

$$A^{-1}\hat{X} = \hat{U} - \mathbf{v}_0\mathbf{1}^\top,$$

where $\mathbf{v}_0$ is a vertex of V. Without loss of generality assume $\mathbf{v}_0 = 0$. Under the perturbation this system becomes

$$\left(A^{-1} + \Delta A^{-1}\right)\left(\hat{X} + \Delta\hat{X}\right) = \hat{U} + \Delta\hat{U}.$$

It is well known that if $\|A\| \|\Delta A^{-1}\| < 1$ then the change in $\hat{X}$ satisfies [45]

$$\left\|\Delta \hat{X}\right\| \leq \frac{\|A\| \left(\|\Delta A^{-1}\| \left\|\hat{X}\right\| + \left\|\Delta \hat{U}\right\|\right)}{1 - \kappa(A) \|\Delta A^{-1}\| / \|A^{-1}\|}.$$

Lemma 4.4 provides that the change in area due to shifts in intersections is of the same magnitude as the change in the numerator of q_0. This is proportional to $L \left\|\Delta \hat{X}\right\|$. The change in area due to shifts in vertices of X is equal to the magnitude of the shifts times the diameter of Y which is $\mathcal{O}(1)$. Thus, the change in area of the intersection is $\mathcal{O}\left((1+L)\left\|\Delta \hat{X}\right\|\right)$.

Corollary 4.8 gives the list of possible representations. The shape of the polygon of intersection is continuous with respect to change in any given vertex or intersection since the graph rewrites of Figure 4.16 map these representations to themselves. Any shift in position is accounted for in Figure 4.17. Therefore, the change in area of the intersection remains $\mathcal{O}\left((1+L)\left\|\Delta \hat{X}\right\|\right)$.

The portions of $\hat{X}$ and $\Delta \hat{X}$ that lie inside Y must be transformed back into the coordinate system of U and V. The area of the polygon due to $\hat{X}$ is at most $\mathcal{O}(1)$. The perturbation of A^{-1} introduces a multiplication of this area by $\|\Delta A^{-1}\|$. This leads to the added term $\mathcal{O}(\Delta V)$ in the statement of the theorem.

The area due to $\Delta \hat{X}$ is multiplied by a constant proportional to $\|A^{-1}\|$. The change in area is then on the order of

$$\|A^{-1}\| (1+L) \left\|\Delta \hat{X}\right\| \leq \frac{\|A^{-1}\| \|A\| (1+L)}{1 - \kappa(A) \|\Delta A^{-1}\| / \|A^{-1}\|} \left(\|\Delta A^{-1}\| \left\|\hat{X}\right\| + \left\|\Delta \hat{U}\right\|\right).$$

Given that $\|A^{-1}\| \|A\| = \kappa(A)$, $\left\|\hat{X}\right\| \leq L$, $\|A^{-1}\| = \mathcal{O}(V)$ and $\|\Delta A^{-1}\| = \mathcal{O}(\Delta V)$ one arrives at the equation in the statement of the theorem. ■

If L, $\kappa(A)$ and $\mathcal{O}(\Delta V/V)$ are sufficiently small then PANG2 is backward stable. This is true if U is not significantly larger than V, the lengths of the edges of V differ by no more than a few orders of magnitude and the relative error in V is reasonable.

4.9 Comparison of algorithms

We present another example problem to show that replacing the triangle intersection algorithm in PANG with the proposal presented here makes it robust. This example was motivated by discussions with Frédéric Hecht at the CIRM Workshop on Parallel Solution Methods for Systems Arising from PDEs, September 16-20, 2019. It systematically generates failures of the triangle intersection subroutines in PANG to successfully compute all intersections to within an error on the order of machine precision.

The example considers a triangulation $\mathbb{T}$ with n identical triangles arranged radially around the origin. To define $\mathbb{T}$ we must define the grid points t_i for $i = 0, ..., n$ and the triangles T_i for $i = 0, ..., n-1$:

$$t_i = \left(\cos\left(\frac{i2\pi}{n}\right), \sin\left(\frac{i2\pi}{n}\right)\right), \; T_i = (0, t_i, t_{i+1}) \implies \mathbb{T} = \{T_i\}_{i=0}^{n-1}.$$

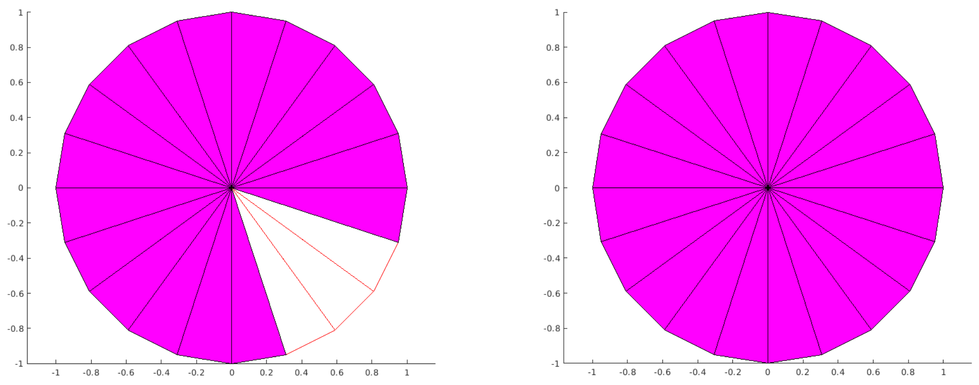


Figure 4.19: Intersection of $\mathbb{T}$ and $\mathbb{T}_\epsilon$ for $\epsilon = 1e - 16$ and $n = 20$. The magenta triangles represent calculated intersections.

This grid is overlaid with a perturbation of itself, $\mathbb{T}_\epsilon$. The point at the origin is shifted by ϵ in a random direction and the radial points are collectively rotated by $\mathcal{O}(\epsilon)$ radians:

$$\tilde{t}_i = \left(\cos\left(\frac{i2\pi}{n} + r_1\right), \sin\left(\frac{i2\pi}{n} + r_1\right)\right), \; \tilde{T}_i = (\epsilon r_2, \tilde{t}_i, \tilde{t}_{i+1}) \implies \mathbb{T}_\epsilon = \left\{\tilde{T}_i\right\}_{i=0}^{n-1},$$

where $|r_1| \leq \epsilon 2\pi/n$ and r_2 is a unit vector in a random direction.

For large enough ϵ errors in consistency are avoided: All subroutines of PANG agree on the underlying geometry of the problem. As ϵ becomes smaller, the triangulations become nearly identical. A small discrepancy between two steps of the triangle intersection algorithm in PANG has already been shown to cause large errors. The overlap of these triangulations presents several such discrepancies.

Figure 4.19 shows the resulting intersections between $\mathbb{T}$ and $\mathbb{T}_\epsilon$ for $\epsilon = 1e - 16$ and $n = 20$. The original triangle intersection algorithm in PANG fails to find the intersection between T_i and $\tilde{T}_i$ for $i = 16$, 17 and 18. The proposed algorithm finds all intersections between T_i and $\tilde{T}_i$. Any remaining intersections (which cannot be independently verified) have area $\mathcal{O}(\epsilon)$.

4.9.1 Comparison of accuracy and computation time

While comparisons of robustness with other polygon clipping algorithms proves challenging and their inclusion in projection algorithms complicated, we can easily compare accuracy and computation time on a single pair of triangles. Let V be the clipping triangle with two sides equal to 1 and the angle between these sides equal to α. Let U, the subject triangle, be an equilateral triangle whose vertices are all 0.5 from the vertex of V with angle α, one of which is halfway between the edges of V with equal length. The coordinates of the vertices as functions of α are presented in Table 4.5.

The exact area of the triangle intersection is straightforward to calculate. First, the portion of U that lies within V is symmetric about the line with angle $\alpha/2$. Thus, we need only consider half of this portion. Given that U is equilateral, this half has one side equal to 0.5, with the adjacent angles equal to $\alpha/2$ and $\pi/6$. This establishes the third angle as $\pi/2 - \pi/6 - \alpha/2$.

V			$2 * U$		
0	0	$\cos(\alpha)$	$\cos(\alpha/2)$	$\cos(\alpha/2+2\pi/3)$	$\cos(\alpha/2+4\pi/3)$
0	1	$\sin(\alpha)$	$\sin(\alpha/2)$	$\sin(\alpha/2+2\pi/3)$	$\sin(\alpha/2+4\pi/3)$

Table 4.5: Coordinates of the clipping and subject triangles. The coordinates of U must be multiplied by 0.5.

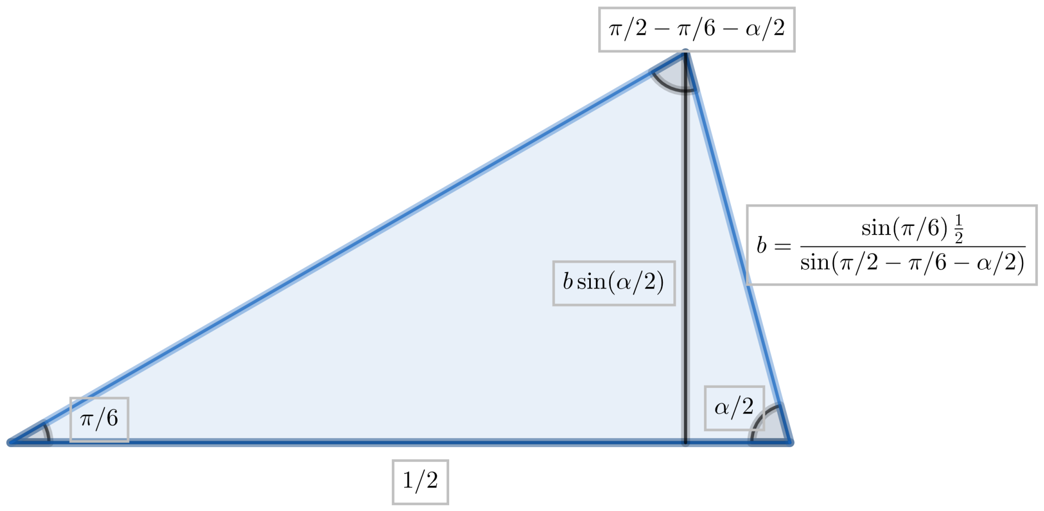

$\pi/2-\pi/6-\alpha/2$
$b = \frac{\sin(\pi/6)\frac{1}{2}}{\sin(\pi/2-\pi/6-\alpha/2)}$
$b\sin(\alpha/2)$
$\pi/6$
$\alpha/2$
$1/2$

Figure 4.20: The half portion of the intersection used to find its area.

Using the law of sines, the other side connected to the angle $\alpha/2$ is equal to

$$b = \frac{1}{2}\frac{\sin(\pi/6)}{\sin(\pi/2-\pi/6-\alpha/2)}.$$

If the base of the triangle is that side equal to $1/2$ then the height is $b\sin(\alpha/2)$. The area of the half portion is then $b\sin(\alpha/2)/4$, or $b\sin(\alpha/2)/2$ for the whole intersection.

For large enough α, part of U comes out the other side of V. Along the ray with angle $\alpha/2$, where the vertex of U is aligned, V intersects at a distance $\cos(\alpha/2)$ from the origin. The portion of U on the other side of V is again symmetric. One half of it is a right angle triangle with height $0.5-\cos(\alpha/2)$ whose adjacent angle is $\pi/6$. Thus, the area of the portion of U to be removed from the intersection is

$$\left(\frac{1}{2}-\cos(\alpha/2)\right)^2\tan(\pi/6). \tag{4.3}$$

The total area of the intersection is then

$$\begin{cases} \frac{1}{4}\frac{\sin(\pi/6)\sin(\alpha/2)}{\sin(\pi/2-\pi/6-\alpha/2)} & 1/2 < \cos(\alpha/2) \\ \frac{1}{4}\frac{\sin(\pi/6)\sin(\alpha/2)}{\sin(\pi/2-\pi/6-\alpha/2)} - \left(\frac{1}{2}-\cos(\alpha/2)\right)^2\tan(\pi/6) & 1/2 \geq \cos(\alpha/2). \end{cases} \tag{4.4}$$

We compare four triangle-triangle intersection algorithms: the original PANG [20]; PANG2, with both change of coordinates and reference-free parametrization and;

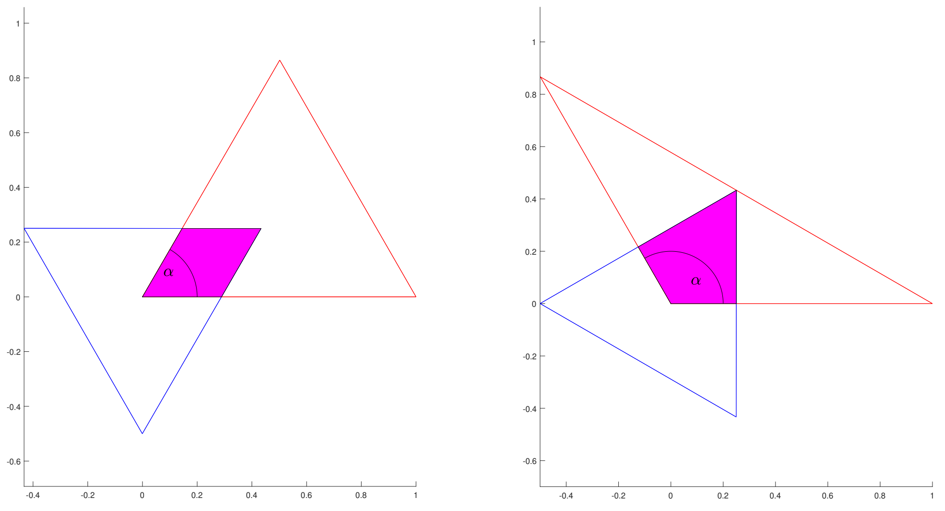


Figure 4.21: V (red) and U (blue) for $\alpha = \pi/3$ (left) and $\alpha = 2\pi/3$ (right).

Sutherland-Hodgman, an established polygon clipping algorithm [10]. The implementations of PANG2 may be found in [31].

Figure 4.22 shows the results for α between 0 and π. Note that the error is striated, with each level associated with a particular fraction of machine epsilon. A black line is drawn along an absolute error of $\epsilon_m/16$. In terms of accuracy, PANG appears to have trouble with small angles, while the reference-free parametrization is nominally worse for midrange angles. In terms of computation time, this implementation of the reference-free parametrization is marginally faster than the change of coordinates. Both versions of PANG2 are faster than their competitors, in particular by nearly an order of magnitude over the more established Sutherland-Hodgman algorithm.

4.10 Conclusions

This chapter has presented an algorithm for the computation of the intersection between two triangles. It includes a proof of its robustness. Specifically, the algorithm is proven to be continuous with respect to input. Under a small error in some calculation, the error in area remains small.

The robustness is ultimately provided by the principle of parsimony. The smallest number of calculations are done so as to ensure that no two produce inconsistent results. For each calculation, the maximum amount of information is extracted and never again sought by separate calculations.

In the following two chapters this algorithm is extended, first to 3D and the intersection of tetrahedra and then to arbitrary dimension and the intersection of simplices.

4.11 Invalid graph configurations

The proof of Lemma 4.6 requires drawing thirteen graphs to determine which satisfy all conditions of an intersection graph. Ten of these graphs are found in Figure 4.13.

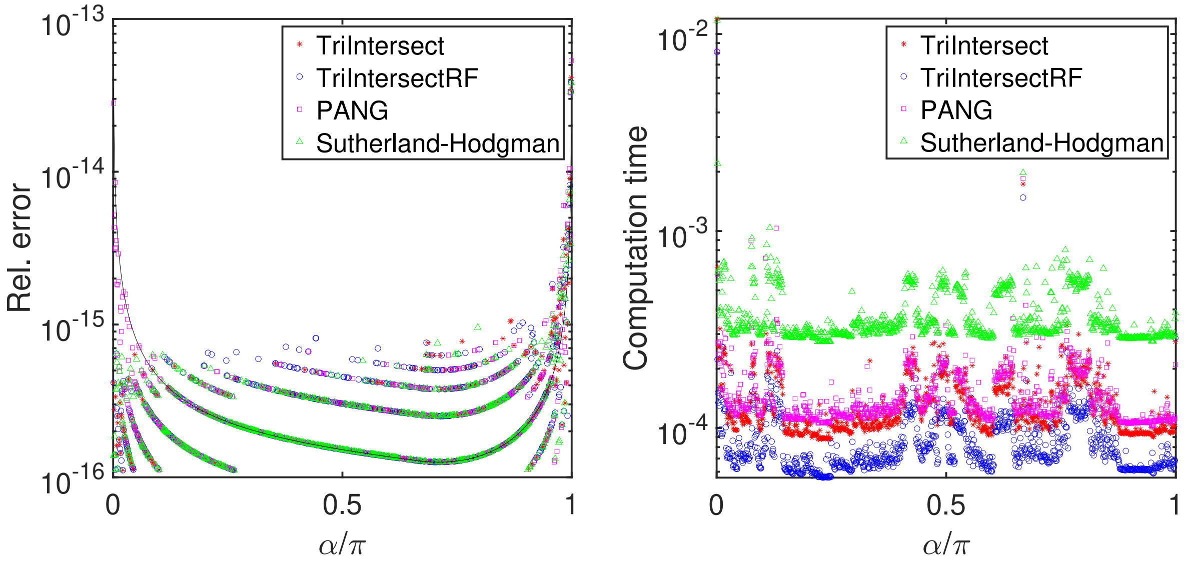


Figure 4.22: Example comparing accuracy and computation time of four triangle-triangle intersection algorithms. Relative error (left) and computation time (right) as a function of α/π.

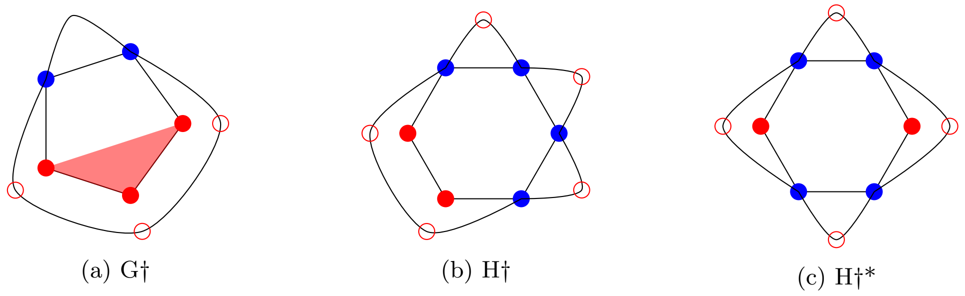


Figure 4.23: The three graphs that do not meet the conditions required of a graph of intersection, see Lemma 4.6.

The three that are found to be invalid are presented in Figure 4.23.

The first of these, G†, from the pairing 2-3, contains an entire triangle in the polygon of intersection. As such, there are no intersections between this triangle and any other. The graph is therefore invalid.

The second two both come from the pairing 4-2. In both, one of the cycles identified by the triangle condition has four triangle vertices. Thus, the cycle does not correspond to a triangle at all, invalidating both graphs.

4.12 Advancing front algorithm

As mentioned in the introduction to this chapter, the intersection algorithm of PANG was paired with an advancing front algorithm. For every pair of intersecting triangles this algorithm tested which neighbours of the two triangles were likely or guaranteed to intersect. Each triangle of one set then received a list of candidate triangles from the second set. Once one triangle of the first set completed its list, a new triangle of the first set was selected and intersected with a triangle from its list of candidates, starting the procedure anew.

One reason PANG failed so frequently in our tests was that this advancing front algorithm would only take as a new candidate pair the last candidate pair it had found. That is, if one triangle finished its list the algorithm would start on a new triangle whose list consisted of only the last triangle determined to intersect it. Thus, if floating-point error caused this calculation to fail then the algorithm would halt abruptly.

As part of this research I introduced a minor fix to this algorithm. Rather than only consider the last candidate triangle found the algorithm now compiles an extensive list for each triangle of the first set. This required using a candidate matrix rather than a candidate vector.

4.13 Additional proofs

Lemma 4.11. *The determinant of* $\begin{bmatrix}\mathbf{w}_i & \mathbf{w}_i^\perp\end{bmatrix}^{-1}\begin{bmatrix}\mathbf{w}_j & \mathbf{w}_j^\perp\end{bmatrix}$ *is positive.*

Proof First note that the determinant of $\begin{bmatrix}\mathbf{w}_i & \mathbf{w}_i^\perp\end{bmatrix}^{-1}$ has the same sign as that of $\begin{bmatrix}\mathbf{w}_i & \mathbf{w}_i^\perp\end{bmatrix}^\top$. Thus, we are concerned with the determinant of

$$\begin{bmatrix}\mathbf{w}_i & \mathbf{w}_i^\perp\end{bmatrix}^\top\begin{bmatrix}\mathbf{w}_j & \mathbf{w}_j^\perp\end{bmatrix} = \begin{bmatrix}\mathbf{w}_i^\top\mathbf{w}_j & \mathbf{w}_i^\top\mathbf{w}_j^\perp \\ (\mathbf{w}_i^\perp)^\top\mathbf{w}_j & (\mathbf{w}_i^\perp)^\top\mathbf{w}_j^\perp\end{bmatrix}.$$

The value of $(\mathbf{w}_i^\perp)^\top\mathbf{w}_j$ is positive by the choice of $\mathbf{w}_i^\perp$ made in Section 4.3.1. The value of $\mathbf{w}_i^\top\mathbf{w}_j^\perp$ is

$$\mathbf{w}_i^\top\mathbf{w}_j^\perp = -(\mathbf{w}_j+\mathbf{w}_k)^\top\mathbf{w}_j^\perp = -\mathbf{w}_k^\top\mathbf{w}_j^\perp$$

which is negative by the same choice. Since $\mathbf{w}_i^\perp$ and $\mathbf{w}_j^\perp$ are the vectors $\mathbf{w}_i$ and $\mathbf{w}_j$ under the same rotation their scalar products are the same. The determinant is then

$$\begin{aligned}\det\left(\begin{bmatrix}\mathbf{w}_i & \mathbf{w}_i^\perp\end{bmatrix}^\top\begin{bmatrix}\mathbf{w}_j & \mathbf{w}_j^\perp\end{bmatrix}\right) &= \begin{vmatrix}\mathbf{w}_i^\top\mathbf{w}_j & \mathbf{w}_i^\top\mathbf{w}_j^\perp \\ (\mathbf{w}_i^\perp)^\top\mathbf{w}_j & (\mathbf{w}_i^\perp)^\top\mathbf{w}_j^\perp\end{vmatrix} \\ &= \left(\mathbf{w}_i^\top\mathbf{w}_j\right)^2 + \mathbf{w}_k^\top\mathbf{w}_j^\perp\mathbf{w}_j^\top\mathbf{w}_i^\perp\end{aligned}$$

which is strictly positive. ■

We use a bound on $\kappa(A)$ given by Hoffmann [26]. This bound requires modification for matrices where $C < 1$. For completeness we prove the form of the bound and a slightly tighter version.

Lemma 4.12. *Let* $\mathbf{v}_1$, $\mathbf{v}_2 \in \mathbb{R}^2$ *such that* α*, the angle between them, is less than* $\pi/2$*, then*

$$\frac{1}{C\sin(\alpha)} \le \kappa\left(\begin{bmatrix}\mathbf{v}_1 & \mathbf{v}_2\end{bmatrix}\right) < \frac{1}{C\sin(\alpha/2)} \tag{4.5}$$

where $C \le 1$ *is the ratio of* $\|\mathbf{v}_1\|$ *to* $\|\mathbf{v}_2\|$.

Proof The matrix $\begin{bmatrix}\mathbf{v}_1 & \mathbf{v}_2\end{bmatrix}$ may be expressed as

$$\begin{bmatrix}\mathbf{v}_1 & \mathbf{v}_2\end{bmatrix} = cR\begin{bmatrix}1 & \cos(\alpha) \\ 0 & \sin(\alpha)\end{bmatrix}\begin{bmatrix}1 & 0 \\ 0 & C\end{bmatrix}$$

where R is a rotation matrix and c is a scalar. The condition number is invariant under rotation and scalar multiplication, and so

$$\kappa\left(\begin{bmatrix}\mathbf{v}_1 & \mathbf{v}_2\end{bmatrix}\right) = \kappa\left(\begin{bmatrix}1 & \cos(\alpha)\\ 0 & \sin(\alpha)\end{bmatrix}\begin{bmatrix}1 & 0\\ 0 & C\end{bmatrix}\right).$$

Since the condition number is based on norms, the upper bound is immediate:

$$\kappa\left(\begin{bmatrix}\mathbf{v}_1 & \mathbf{v}_2\end{bmatrix}\right) \leq \kappa\left(\begin{bmatrix}1 & \cos(\alpha)\\ 0 & \sin(\alpha)\end{bmatrix}\right)\kappa\left(\begin{bmatrix}1 & 0\\ 0 & C\end{bmatrix}\right) = \frac{\cot(\alpha/2)}{C} < \frac{1}{C\sin(\alpha/2)}.$$

To prove the lower bound we must find the limit of the condition number as C approaches 0, then show the condition number cannot fall below this limit.

First we must find the singular values of the matrix:

$$\begin{bmatrix}1 & 0\\ C\cos(\alpha) & C\sin(\alpha)\end{bmatrix}\begin{bmatrix}1 & C\cos(\alpha)\\ 0 & C\sin(\alpha)\end{bmatrix} = \begin{bmatrix}1 & C\cos(\alpha)\\ C\cos(\alpha) & C^2\end{bmatrix},$$

giving the characteristic equation

$$(1-\sigma)(C^2-\sigma) - C^2\cos^2(\alpha) = \sigma^2 - (1+C^2)\sigma - C^2(\cos^2(\alpha)-1)$$

which implies

$$\begin{aligned}
\kappa\left(\begin{bmatrix}\mathbf{v}_1 & \mathbf{v}_2\end{bmatrix}\right)^2 &= \frac{1+C^2+\sqrt{(1+C^2)^2-4C^2\sin^2(\alpha)}}{1+C^2-\sqrt{(1+C^2)^2-4C^2\sin^2(\alpha)}}\\
&= \frac{\left(1+C^2+\sqrt{(1+C^2)^2-4C^2\sin^2(\alpha)}\right)^2}{(1+C^2)^2-(1+C^2)^2+4C^2\sin^2(\alpha)}\\
&= \frac{\left(\frac{1}{C}+C+\sqrt{\left(\frac{1}{C}+C\right)^2-4\sin^2(\alpha)}\right)^2}{4\sin^2(\alpha)}\\
\implies \kappa\left(\begin{bmatrix}\mathbf{v}_1 & \mathbf{v}_2\end{bmatrix}\right) &= \frac{\frac{1}{C}+C+\sqrt{\left(\frac{1}{C}+C\right)^2-4\sin^2(\alpha)}}{2\sin(\alpha)}.
\end{aligned}$$

Take the limit as C approaches 0:

$$\lim_{C\to 0}\kappa\left(\begin{bmatrix}\mathbf{v}_1 & \mathbf{v}_2\end{bmatrix}\right) = \frac{\frac{1}{C}+\sqrt{\frac{1}{C^2}}}{2\sin(\alpha)} = \frac{1}{C\sin(\alpha)}.$$

It remains to prove this is a lower bound of $\kappa\left(\begin{bmatrix}\mathbf{v}_1 & \mathbf{v}_2\end{bmatrix}\right)$. Note that this is equivalent to proving

$$\frac{1}{\sin(\alpha)} \leq C\kappa\left(\begin{bmatrix}\mathbf{v}_1 & \mathbf{v}_2\end{bmatrix}\right) < \frac{1}{\sin(\alpha/2)}.$$

The maximum of $C\kappa\left(\begin{bmatrix}\mathbf{v}_1 & \mathbf{v}_2\end{bmatrix}\right)$ occurs at $C = 1$, and our proposed minimum occurs at $C = 0$. Then it suffices to prove $C\kappa\left(\begin{bmatrix}\mathbf{v}_1 & \mathbf{v}_2\end{bmatrix}\right)$ increases monotonically with C.

Take the derivative of $C\kappa\left(\begin{bmatrix}\mathbf{v}_1 & \mathbf{v}_2\end{bmatrix}\right)$ with respect to C:

$$\begin{aligned}\frac{d}{dC}C\kappa\left(\begin{bmatrix}\mathbf{v}_1 & \mathbf{v}_2\end{bmatrix}\right) =& \frac{d}{dC}\frac{1+C^2+\sqrt{(1+C^2)^2-4C^2\sin^2(\alpha)}}{2\sin(\alpha)} \\ =& \frac{2C+\frac{2(1+C^2)C-4C\sin^2(\alpha)}{\sqrt{(1+C^2)^2-4C^2\sin^2(\alpha)}}}{2\sin(\alpha)} \\ =& \frac{C}{\sin(\alpha)}\left(1+\frac{1+C^2-2\sin^2(\alpha)}{\sqrt{(1+C^2)^2-4C^2\sin^2(\alpha)}}\right).\end{aligned}$$

While C and $\sin(\alpha)$ are positive by assumption, it remains to confirm the second term is as well. To do this, we must compare the magnitudes of the terms in the quotient:

$$\begin{aligned}(1+C^2)^2-4\sin^2(\alpha)(1+C^2)+4\sin^4(\alpha)-(1+C^2)^2+4C^2\sin^2(\alpha)& \\ =&4\sin^4(\alpha)-4\sin^2(\alpha)-4C^2\sin^2(\alpha)+4C^2\sin^2(\alpha) \\ =&4\sin^2(\alpha)\left(\sin^2(\alpha)-1\right) \\ =&-4\sin^2(\alpha)\cos^2(\alpha)<0 \\ \implies 1+C^2-2\sin(\alpha)<&\sqrt{(1+C^2)^2-4C^2\sin^2(\alpha)}\end{aligned}$$

and so the number in parantheses is between 0 and 2. Therefore, $C\kappa\left(\begin{bmatrix}\mathbf{v}_1 & \mathbf{v}_2\end{bmatrix}\right)$ increases monotonically with C. ■

4.14 Apocrypha

Our first version of this algorithm was markedly different to the current one. Most notably, it lacked the pairing of intersections indicated by their numerators from Table 4.3, allowing errors to occur unpaired. Before discovering this pairing, we had implemented a way to patch the problem of unpaired errors. The following sections consider these unpaired intersection errors and their effects.

4.14.1 Unpaired intersection errors

Figure 4.24 gives examples of the possible unpaired intersection errors. While the configurations of the triangles can change slightly the effect of each error is the same. For each, an intersection is translated a distance ϵ along the reference line of Y, finding itself on the other side of one of the other two reference lines that intersects the first.

The effects of the errors will now be explained in full.

(a) The intersection error has caused a Y-in-X error. The calculated intersection now includes the vertex of Y. If the intersection has moved ϵ along the reference line then the new area of the polygon of intersection has increased by $\mathcal{O}(\epsilon)$. The error behaves the same as in Figure 4.8c.

(b) The intersection error has caused a Y-in-X error. The polygon has grown, again by $\mathcal{O}(\epsilon)$. However, the shape of the polygon has not changed: It remains a triangle. In essence, the intersection has been replaced by the vertex of Y.

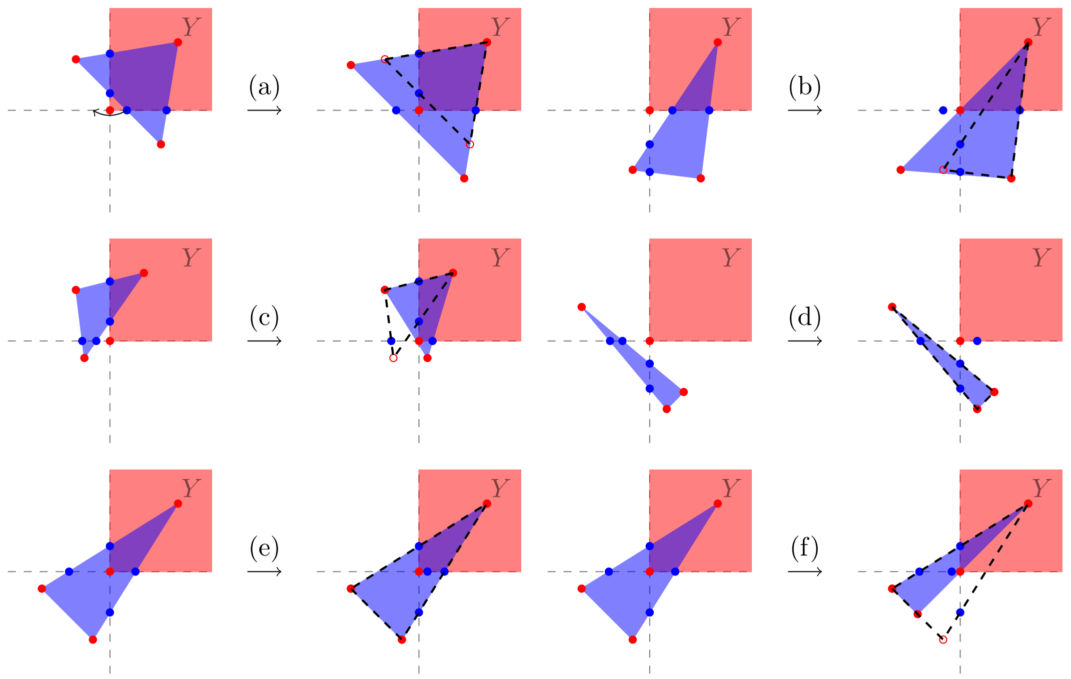


Figure 4.24: Results of a single intersection error over a vertex of Y. The upper right quadrant is a corner of Y. Blue dots are the calculated locations of the intersections. Red dots are the effective locations of the triangle vertices based on the intersections, and the blue region is the effective location of X. Open red circles are the true locations of the vertices of X. The true location of X is outlined with a dashed line.

(c) The intersection error has caused a Y-in-X error. With regards to the shape of the polygon, the behaviour is that of Figure 4.8b. The intersection has again grown by $\mathcal{O}(\epsilon)$.

(d) The intersection error has technically caused a Y-in-X error. However, there are only two points in the polygon and so no intersection is calculated.

(e) No Y-in-X error occurs. The shape and size of the polygon does not change.

(f) No Y-in-X error occurs. An intersection has been removed from the polygon. This is now a degenerate case where the vertex of Y acts as an intersection and will be discussed alongside the other degenerate cases later. The area of the polygon has been reduced by $\mathcal{O}(\epsilon)$.

Errors (b) and (d) can also occur as described without intersections on the other reference line intersecting the vertex of Y.

Figure 4.25 shows the result of two intersection errors that occur over the same vertex but on different reference lines. Two intersection errors over the same vertex of Y occur only if some of the vertices of X sit close to this vertex of Y. Notably they require intersections to be calculated for two reference lines. The triangle X must then be found in at least three of the four quadrants defined by these reference lines.

These compound errors can be much more problematic, as explained here.

(a)(a), (c)(b) There is no practical difference between these errors and those of their single counterparts (a) and (c).

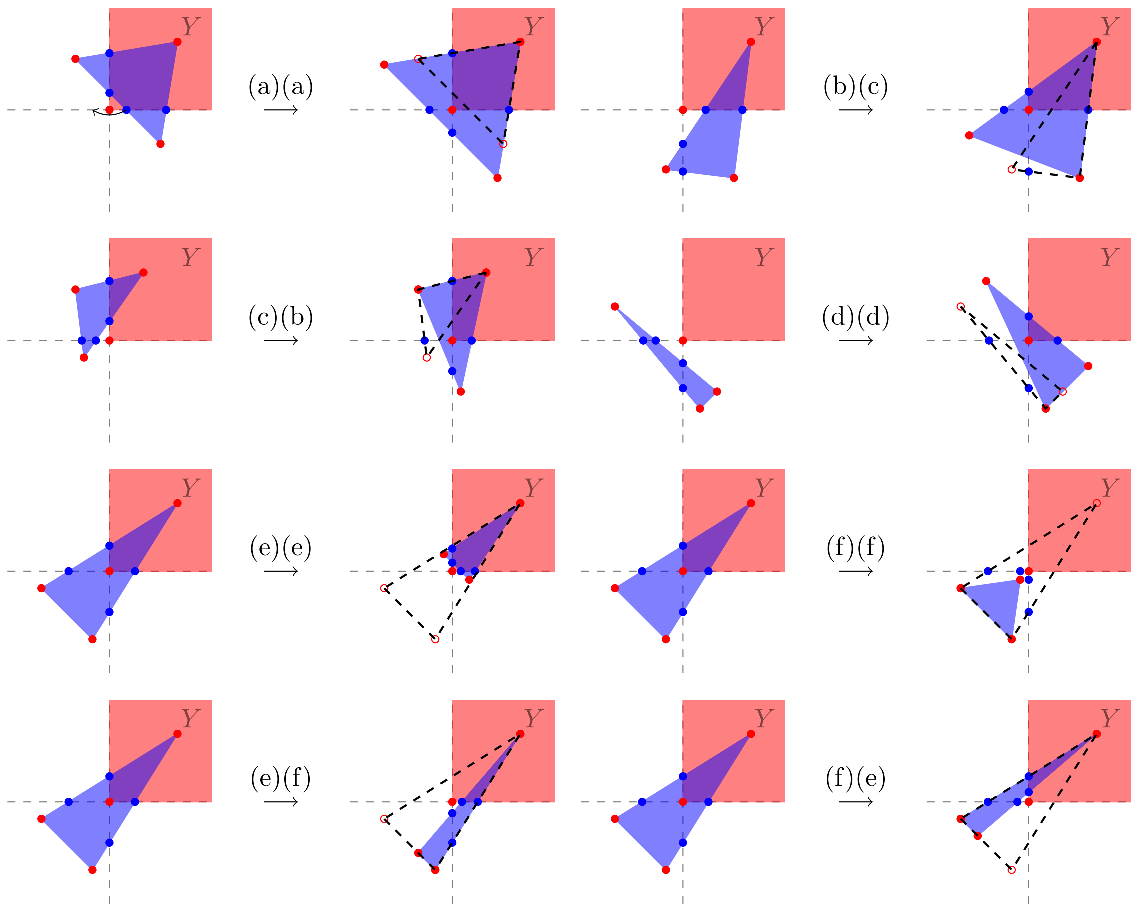


Figure 4.25: Results of two intersection errors over a vertex of Y, occurring over different reference lines. The upper right quadrant is a corner of Y. Blue dots are the calculated locations of the intersections. Red dots are the effective locations of the triangle vertices based on the intersections, and the blue region is the effective location of X. Open red circles are the true locations of the vertices of X. The true location of X is outlined with a dashed line.

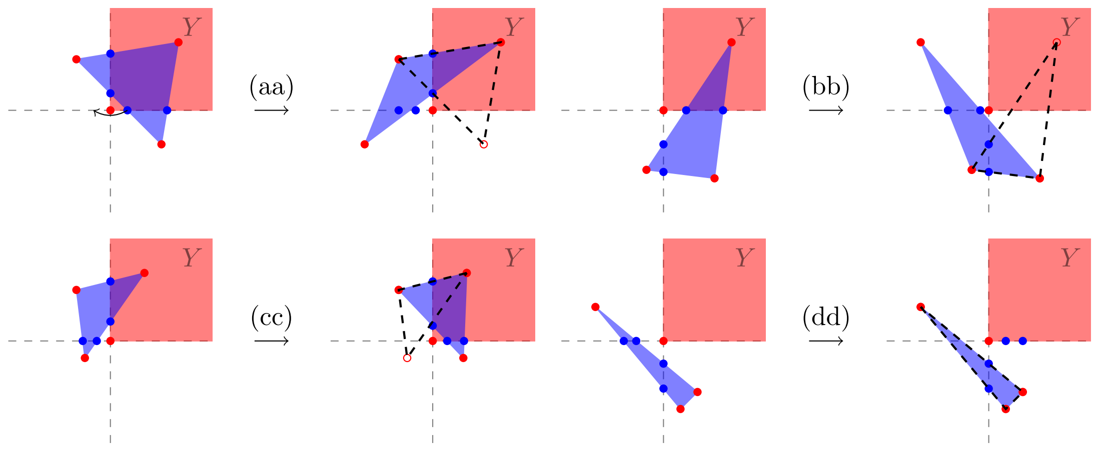


Figure 4.26: Results of two intersection error over a vertex of Y, occurring over the same reference line. The upper right quadrant is a corner of Y. Blue dots are the calculated locations of the intersections. Red dots are the effective locations of the triangle vertices based on the intersections, and the blue region is the effective location of X. Open red circles are the true locations of the vertices of X. The true location of X is outlined with a dashed line.

(b)(c) The polygon of intersection has again grown by $\mathcal{O}(\epsilon)$, where ϵ is the size of the intersection error. The error has the same result as Figure 4.8b on the shape and makeup of the polygon. This error is identical to (c)(b), since the order in which the errors occur is inconsequential.

(d)(d) The Y-in-X error caused by one instance of (d) now causes a change in the polygon with a second instance of (d). The entirely erroneous polygon is a triangle with two sides of $\mathcal{O}(\epsilon)$, meaning it has an area of $\mathcal{O}(\epsilon^2)$.

(e)(e) A second instance of (e) causes a Y-in-X error, with the vertex of Y being removed from the polygon. Because this vertex is replaced in the polygon by two intersections, as in Figure 4.8a, the polygon is only reduced by a triangle with area $\mathcal{O}(\epsilon^2)$.

(f)(f) A second instance of (f) causes a Y-in-X error. The loss of three corners of the polygon of intersection is catastrophic. If, as shown in the figure, there is only one vertex of X within or on the other side of Y then the error in area is $\mathcal{O}(\epsilon)$ and the result can be thought of as a combination of Figures 4.8b and 4.8c. If, on the other hand, there are two vertices of X in this quadrant then the error can be $\mathcal{O}(1)$. It is essential to guard against an error of this kind.

(e)(f), (f)(e) These compound errors cause Y-in-X errors. The result is indistinct from a solitary (f) error, save for an additional $\mathcal{O}(\epsilon)$ loss of area.

If the compound errors occur over the same reference line then the errors resemble those of Figure 4.26. As in the other case, such errors occur only if some of the vertices of X reside near this vertex of Y. However, they do not need intersections to be calculated on the other reference line intersecting this vertex of Y.

These last four compound errors are explained here.

(aa) The Y-in-X error caused by the first instance of (a) is resolved by the second instance, though not in our favour. There is now a deletion in the polygon of

Intersection error	Abs. error	Shape change	Compound error	Abs. error	Shape change
(a), (a)(a)	$\mathcal{O}(\epsilon)$	4.8c	(d)(d)	$\mathcal{O}(\epsilon^2)$	4.8a
(b)	$\mathcal{O}(\epsilon)$		(e)(e)	$-\mathcal{O}(\epsilon^2)$	4.8a
(c), (c)(b), (b)(c)	$\mathcal{O}(\epsilon)$	4.8b	(f)(f)	$-\mathcal{O}(1)$	4.8b + 4.8c*
(d), (dd)	0		(aa)	$-\mathcal{O}(1)$	4.8c + 4.8b
(e)	0		(bb)	$-\mathcal{O}(1)$	4.8c*
(f), (e)(f), (f)(e)	$-\mathcal{O}(\epsilon)$	4.8b	(cc)	$\mathcal{O}(\epsilon)$	4.8b + 4.8a

Table 4.6: All possible intersection errors, the absolute error in the area of the polygon of intersection, and equivalent change in shape found in Figure 4.8. A minus sign in an error indicates a decrease in area, whereas the lack of one indicates an increase in area.

intersection. If there is a vertex of X in the quadrant containing Y, as shown, the error in area can be as large as $\mathcal{O}(1)$. With regards to shape, this compound error is equivalent to a combination of Figures 4.8c and 4.8b.

(bb) As in the case of the compound error (f)(f), how problematic this error is depends on the position of the vertices of X. If there is only one vertex of X within the quadrant containing Y then the polygon of intersection is deleted, though it possessed an area of $\mathcal{O}(\epsilon)$. If there are two vertices of X in this quadrant then the loss of area may be of size $\mathcal{O}(1)$.

(cc) This is in some sense the reverse of (aa). This is distinct from (c) only in that there is a triangle of area $\mathcal{O}(\epsilon)$ missing from the polygon. The shape has undergone the effects of Figures 4.8b and 4.8a.

This error does not cause as dramatic a change in area as (aa) since it falls within a triangle whose base lies between the starting positions of the displaced intersections and their final positions, which is at most ϵ in length. The area deleted in (aa) is defined instead by the length between the vertex of Y and the closest intersection along the vertical reference line, which is not restricted to ϵ.

(dd) This error is indistinct from that of a single instance of (d).

Errors (bb) and (dd) can occur without intersections on the other reference line intersecting this vertex of Y.

The list of intersection errors are tabulated, along with the absolute error and effect of shape on the polygon of intersection, in Table 4.6. Proofs of the magnitude of the absolute error can be found in Appendix 4.14.3. One may ask also for the results of three or four intersection errors over the same vertex of Y. However, note that the final results of all compound errors are valid configurations of the intersections. One can then combine the results listed here twice over to determine the overall effect.

For example, suppose one wants to subject a polygon of intersection to two counts of (c) error along one reference line and two counts of (a) error on the other. Since the compound error (cc) returns the starting configuration for the error (aa), one need only draw a line connecting the starting configuration of (cc) to the final position of (aa). In this way, all intersection errors are exhausted.

It is clear from this analysis that there are three intersection errors to be concerned about: (f)(f), (aa) and (bb). Note that all of these errors have aspects of their configurations in common. Along the horizontal reference line the two intersections fall outside of the triangle Y. The configuration of the two intersections along the other reference line then varies. Figure 4.27 shows the four possible configurations of two

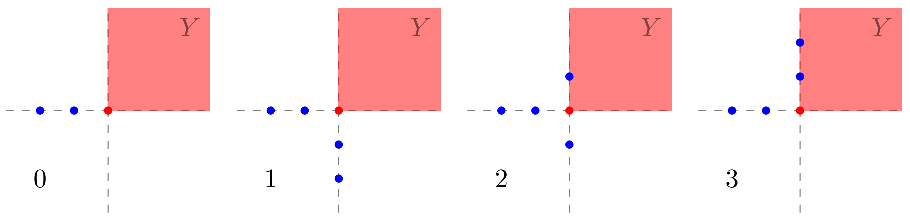


Figure 4.27: The four configurations when two intersections fall outside of Y along the horizontal reference line. Each is numbered for reference.

Configuration	Starting position before error	Final position after error
0	(d)	(bb)
1	(d)	(bb), (f)(f)
2		(b), (f)
3	(b), (c)	(e)(f), (f)(e), (aa), (dd)

Table 4.7: Correspondence between the configurations of Figure 4.27 and starting and final positions of the errors of Figures 4.24, 4.25 and 4.26.

intersections around a vertex of Y. Table 4.7 matches each of these configurations with a starting position from Figure 4.24 and final positions after errors.

Errors (bb) and (f)(f) are only problematic when two vertices of X are found in the same quadrant as Y. Adding the vertex of Y to the polygon of intersection in these configurations bounds the error in area as $\mathcal{O}(\epsilon)$. Moreover, adding this vertex to the polygon for the starting position of (d) does not affect the area or shape of the polygon. In fact, it is equivalent to an error of type (d). Thus, if configurations 0 or 1 are encountered during the algorithm then the vertex of Y should be considered to be within X.

Configuration 2 is the result of errors (b) or (f). Neither error is found to be problematic and so configuration 2 requires no additional changes to the algorithm.

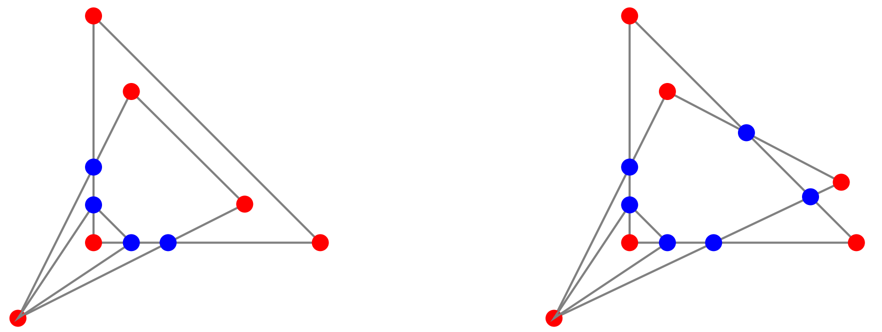
For configuration 3 adding the vertex of Y to the polygon to the starting positions of (b) or (c) will cause a type (b) error. However, doing so would restrict errors (e)(f) and (f)(e) to single (f) errors. It is therefore necessary to identify if configuration 3 belongs to these starting positions or to the final positions of (e)(f), (f)(e) or (aa) errors. The errors (e)(f) and (f)(e) are not problematic so we focus on identifying (aa) errors.

The defining difference between the final position of (aa) and the starting position of (c) is the presence of a vertex of X in the lower right quadrant. Such a vertex clearly shows the configuration is erroneous as the vertex of Y somehow lies within the triangle formed by the points of intersection and this vertex of X yet is not within X. Adding the vertex of Y back into the polygon restricts the error in area of (aa) and some cases of (e)(f) and (f)(e) and has no effect on (dd).

To guard against the three problematic errors the algorithm must now identify which, if any, of the four configurations has occurred and, in the case of configuration 3, the relative quadrants of the vertices of X.

4.14.2 Graphs created through unpaired intersection errors

Corollary 4.8 states that Figure 4.13 an exhaustive list of representations. However, if intersection errors are allowed to be unpaired, then this is no longer true as there is an



(a) H†, resulting from graph G under an (e)(e) error. (b) J†, resulting from graph H under an (e)(e) error.

Figure 4.28: The two graphs that can result from (e)(e) errors not already considered.

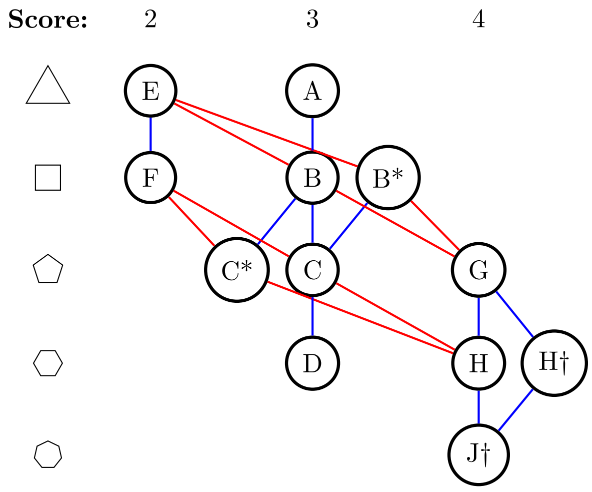


Figure 4.29: Graph of intersection connectivity when unpaired intersection errors are considered. This is Figure 4.17 with the addition of graphs H† and J†.

implicit restriction that the graph be convex. The intersection error (e)(e), see Figure 4.25, can create a hole in these graphs.

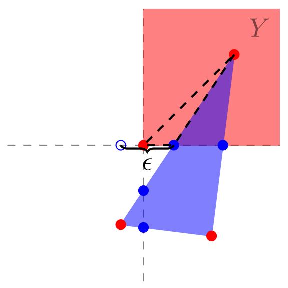
Figure 4.28 shows the algorithmic graphs caused by (e)(e) errors. Graph G under an (e)(e) error gives the graph H†, presented as impossible in Figure 4.23b. The configuration in Figure 4.28a combines two vertices to make it possible. Graph H under the same error gives J†, not considered as it violates restriction (iv) by having 7 corners in the polygon of intersection. The error in area is small, as explained in Sections 4.14.1 and 4.14.3.

4.14.3 Error in area due to intersection errors

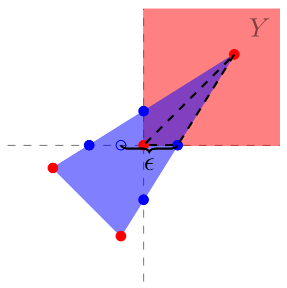
In the following lemmas the intersection errors occur due to one or two displacements of points of intersection by ϵ along a given reference line of Y.

Lemma 4.13. *Errors (a), (b), (c), (a)(a), (b)(c), (c)(b) and (c)(c) add an area of $\mathcal{O}(\epsilon)$ to the polygon of intersection.*

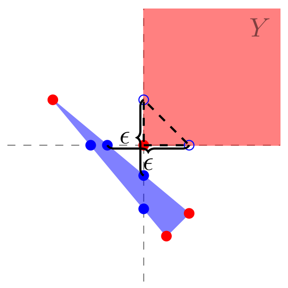
Proof Figure 4.30a shows an example of an error of type (b) adding a triangle to the polygon of intersection. Since the largest distance the corner of the polygon can move



(a) Addition of a triangle with base less than ϵ to the polygon of intersection.



(b) Removal of a triangle with base less than ϵ from the polygon of intersection.



(c) Addition of a triangle with base and height less than ϵ to the polygon of intersection.

Figure 4.30: Displacements of intersections of size $\mathcal{O}(\epsilon)$ cause changes in area of $\mathcal{O}(\epsilon)$ or smaller.

is ϵ by assumption the triangle has a base of at most ϵ. Regardless of the triangle's height it therefore has area $\mathcal{O}(\epsilon)$.

This principle applies to each of the errors listed in the statement of the lemma. The only changes are the shape and composition of the polygon of intersection and the direction of the shift by ϵ. ■

Note that Lemma 4.13 applies also to errors of type (aa) when there are no vertices of X in the quadrant that contains Y.

Lemma 4.14. *Errors (f), (e)(f) and (f)(e) remove an area of $\mathcal{O}(\epsilon)$ from the polygon of intersection. Additionally, if there is only one vertex of X in the quadrant that contains Y then errors (f)(f) and (bb) remove an area of $\mathcal{O}(\epsilon)$.*

Proof Figure 4.30b shows an example of an error of type (f) removing a triangle from the polygon of intersection. As in Lemma 4.13 the corner of the polygon can move at most ϵ and so the triangle being removed has area $\mathcal{O}(\epsilon)$.

This movement occurs twice for errors (e)(f) and (f)(e), resulting again in an error of $\mathcal{O}(\epsilon)$, albeit twice the size in magnitude.

If there is only one vertex of X in the quadrant that contains Y, either inside Y or across the third reference line, then each instance of (f) in a (f)(f) error deletes a triangle of area $\mathcal{O}(\epsilon)$. Since the polygon of intersection in this case is composed of two such triangles this error represents a deletion of the polygon in its entirety. The absolute error in area remains $\mathcal{O}(\epsilon)$.

Under the same assumptions for a (bb) error the polygon is a single triangle with base at most ϵ. It has an area $\mathcal{O}(\epsilon)$ and so while it is deleted by a (bb) error the absolute error in area is $\mathcal{O}(\epsilon)$. ■

Lemma 4.15. *Error (d)(d) adds an area of $\mathcal{O}(\epsilon^2)$ to the polygon of intersection. Error (e)(e) removes an area of $\mathcal{O}(\epsilon^2)$ from the polygon.*

Proof Figure 4.30c shows an example of a (d)(d) error where two intersections have been displaced by ϵ over the vertex of Y. In this case the base and height of the triangle being added to the polygon of intersection, which is the entire polygon, has area $\mathcal{O}(\epsilon^2)$.

These displacements happen in reverse for (e)(e) errors. This triangle is therefore removed from the polygon for such errors. ■

Chapter 5

Intersection of tetrahedra

In which one extends the previous algorithm to the 3D case.

5.1 Introduction

The previous chapter explored a robust algorithm for the calculation of the intersection between two triangles, named PANG2. It is natural to ask for its extension to 3D. That is, what is the equivalent robust algorithm for the calculation of intersections between two tetrahedra? We shall refer to such an algorithm as PANG2-3D.

Many aspects of PANG2 are straightforward to generalize to a higher dimension, such as the change of coordinates. But there are now new types of intersections, those between edges of X and faces of Y and those between faces of X and edges of Y. As well, how does one deal with the vertices of one tetrahedra that lie within the other?

Throughout this chapter I will show how to make the extensions from triangles to tetrahedra. In doing so, extensions to higher dimensions will become obvious, which will lead to Chapter 6.

5.2 Change of coordinates

We consider two tetrahedra U and V with intersection W, all of which lie within $\mathbb{R}^3$. To simplify calculations, transform the tetrahedron V into the reference tetrahedron Y with vertices at the positions $(0,0,0)$, $(1,0,0)$, $(0,1,0)$ and $(0,0,1)$. The tetrahedron U is likewise transformed under the same affine transformation into the tetrahedron X. To do so, one must determine the nature of the affine transformation.

Represent the positions of the vertices of V by the matrix

$$\mathbf{v}_0\mathbf{1}^\top + \begin{bmatrix}\mathbf{0} & \mathbf{v}_1 & \mathbf{v}_2 & \mathbf{v}_3\end{bmatrix},$$

where $\mathbf{v}_0$ is the position of the vertex to be mapped to the origin, $\mathbf{v}_1$, $\mathbf{v}_2$ and $\mathbf{v}_3$ are the vectors leading between $\mathbf{v}_0$ and the remaining vertices, $\mathbf{0}$ is a column vector with three zeros and $\mathbf{1}^\top = \begin{bmatrix}1 & 1 & 1 & 1\end{bmatrix}$. Ideally, $\mathbf{v}_1$, $\mathbf{v}_2$ and $\mathbf{v}_3$ are orthogonal. The best choice of $\mathbf{v}_0$ is one in which this is true, or nearly so.

The process of transforming from the vertices of V to the vertices of Y can be written as an affine transformation:

$$A\left(\mathbf{v}_0\mathbf{1}^\top + \begin{bmatrix}\mathbf{0} & \mathbf{v}_1 & \mathbf{v}_2 & \mathbf{v}_3\end{bmatrix}\right) + \mathbf{b}\mathbf{1}^\top = \begin{bmatrix}0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}.$$

The vector $\mathbf{b}$ is then $-A\mathbf{v}_0$ and the matrix A is the inverse of the matrix $\begin{bmatrix}\mathbf{v}_1 & \mathbf{v}_2 & \mathbf{v}_3\end{bmatrix}$.

This affine transformation must be applied to the 'subject' tetrahedron U to acquire its transformation X. As with V, the position of the vertices of U may be represented by the matrix $\mathbf{u}_0\mathbf{1}^\top + \begin{bmatrix}\mathbf{0} & \mathbf{u}_1 & \mathbf{u}_2 & \mathbf{u}_3\end{bmatrix}$. Let the i–th vertex of X have position (x_i, y_i, z_i). These values may then be found by solving the system

$$\begin{bmatrix}\mathbf{v}_1 & \mathbf{v}_2 & \mathbf{v}_3\end{bmatrix}\begin{bmatrix}x_1 & x_2 & x_3 & x_4\\ y_1 & y_2 & y_3 & y_4\\ z_1 & z_2 & z_3 & z_4\end{bmatrix} = \mathbf{u}_0\mathbf{1}^\top + \begin{bmatrix}\mathbf{0} & \mathbf{u}_1 & \mathbf{u}_2 & \mathbf{u}_3\end{bmatrix} - \mathbf{v}_0\mathbf{1}^\top.$$

5.3 Corners of the intersection

The intersection between the tetrahedra X and Y is a polyhedron Z. There are four types of corners to this polyhedron: vertices of X that lie inside Y; intersections between the edges of X and the faces of Y; intersections between the faces of X and the edges of Y and; vertices of Y that lie inside X. These corners form a hierarchy, with each type informing the calculations of later types. The levels of this hierarchy will be considered one at a time.

5.3.1 Vertices of X that lie inside Y

The reference tetrahedron Y is bounded by four infinite planes: $P_x = \{x = 0\}$, $P_y = \{y = 0\}$, $P_z = \{z = 0\}$ and $P_{xyz} = \{x + y + z = 1\}$. Each plane P_γ defines a parameter $p_\gamma(\mathbf{x})$ that is positive or zero when the point $\mathbf{x} = (x, y, z) \in Y$ and negative otherwise. For three of these planes this parameter is one of the coordinates, $p_\gamma = \gamma$ for $\gamma \in \{x, y, z\}$. For the fourth plane, $p_{xyz}(\mathbf{x}) = 1 - x - y - z$. The i–th vertex of X, $\mathbf{x}_i = (x_i, y_i, z_i)$, lies in Y if and only if $\text{sign}(p_\gamma(\mathbf{x}_i)) = 1$ for all γ. The function $\text{sign}(p)$ is found in equation (4.1).

The signs of $p_\gamma(\mathbf{x}_i)$ indicate the number of intersections between the edges of X and the plane P_γ to calculate. For example, the edge between the i–th and j–th vertices of X intersects P_γ only if $\text{sign}(p_\gamma(\mathbf{x}_i)) \neq \text{sign}(p_\gamma(\mathbf{x}_j))$, assuming neither value of p_γ is equal to zero. The case where $p_\gamma(\mathbf{x}_i) = 0$ is considered in [31] and will be briefly summarized here. Moving $\mathbf{x}_i$ an imperceptible distance into Y does not change the shape of the polyhedron of intersection. Thus, the degenerate case where $p_\gamma(\mathbf{x}_i) = 0$ can be treated as the non-degenerate case where $p_\gamma(\mathbf{x}_i) = \epsilon/2$. It is therefore practical to use the binary-valued sign function previously defined.

Proposition 5.1. *Only 0, 3 or 4 intersections may occur between the edges of X and the plane P_γ.*

Proof For an intersection to exist, $\text{sign}(p_\gamma(\mathbf{x}_i))$ and $\text{sign}(p_\gamma(\mathbf{x}_j))$ must disagree. There are four $p_\gamma(\mathbf{x}_i)$ $(i = 1, ..., 4)$, and $\text{sign}(p_\gamma(\mathbf{x}_i))$ may take one of two values. There are only three ways to partition four objects $(p_\gamma(\mathbf{x}_i))$ into two groups (either 0 or 1), which may be proven by the partition function. These partitionings are listed in Table 5.1,

$m(a)$	$m(b)$	pairs
4	0	0
3	1	3
2	2	4

Table 5.1: Ways to partition four elements into two parts.

Parameters	$x=0$	$y=0$	$z=0$	$x+y+z=1$
p_γ	x	y	z	$1-x-y-z$
q_γ	y	z	x	x
r_γ	z	x	y	y

Table 5.2: Parameterizations of the point (x, y, z) for the given plane.

where $m(a)$ and $m(b)$ are the multiplicities of elements labelled a and b, respectively. A pair is formed by taking one element of each group. The number of pairs is then the product of the two multiplicities. ■

This proposition tells us that the part of X that intersects the plane of Y is a triangle, a quadrilateral, or does not exist. This allows us to consider the intersection of these shapes with the face of Y that lies in the plane.

5.3.2 Intersections between edges of X and faces of Y

Suppose $\operatorname{sign}(p_\gamma(\mathbf{x}_i)) \neq \operatorname{sign}(p_\gamma(\mathbf{x}_j))$ for some γ. Then there is an intersection between the edge of X lying between the i–th and j–th vertices and the plane P_γ. This intersection lies in the plane P_γ and so its value of p_γ is zero. There remain two coordinates needed to ascertain its position in $\mathbb{R}^3$.

We parametrize the plane P_γ with the coordinates q_γ and r_γ. These are chosen such that the face of Y lies between the lines $q_\gamma = 0$, $r_\gamma = 0$ and $q_\gamma + r_\gamma = 1$. They are listed in Table 5.2.

This table is the natural extension of Table 4.2 from 2D to 3D. We require an additional row and an additional column to accommodate the third dimension. Recall in 2D the two parameters p and q were chosen to increase orthogonally, including for the line $x + y = 1$. In 3D this is eschewed so that the parameters stay aligned with the edges of the tetrahedron Y.

The intersection between P_γ and the edge between the i–th and j–th vertices of X,

Plane P	Numerator of q_γ^{ij}	Numerator of r_γ^{ij}	Numerator of $1-q_\gamma^{ij}-r_\gamma^{ij}$
$x=0$	$\begin{vmatrix} x_i & y_i \\ x_j & y_j \end{vmatrix}$	$\begin{vmatrix} x_i & z_i \\ x_j & z_j \end{vmatrix}$	$\begin{vmatrix} x_i & 1-y_i-z_i \\ x_j & 1-y_j-z_j \end{vmatrix}$
$y=0$	$\begin{vmatrix} y_i & z_i \\ y_j & z_j \end{vmatrix}$	$\begin{vmatrix} y_i & x_i \\ y_j & x_j \end{vmatrix}$	$\begin{vmatrix} y_i & 1-x_i-z_i \\ y_j & 1-x_j-z_j \end{vmatrix}$
$z=0$	$\begin{vmatrix} z_i & x_i \\ z_j & x_j \end{vmatrix}$	$\begin{vmatrix} z_i & y_i \\ z_j & y_j \end{vmatrix}$	$\begin{vmatrix} z_i & 1-x_i-y_i \\ z_j & 1-x_j-y_j \end{vmatrix}$
$x+y+z=1$	$\begin{vmatrix} 1-y_i-z_i & x_i \\ 1-y_j-z_j & x_j \end{vmatrix}$	$\begin{vmatrix} 1-x_i-z_i & y_i \\ 1-x_j-z_j & y_j \end{vmatrix}$	$\begin{vmatrix} 1-x_i-y_i & z_i \\ 1-x_j-y_j & z_j \end{vmatrix}$

Table 5.3: Numerators of the relevant values for each plane of Y.

Face	$\mathbf{v}_1$	$\mathbf{v}_2$	$\mathbf{v}_3$
$x = 0$		q_x^{ij}	r_x^{ij}
$y = 0$	r_y^{ij}		q_y^{ij}
$z = 0$	q_z^{ij}	r_z^{ij}	
$x + y + z = 1$	q_{xyz}^{ij}	r_{xyz}^{ij}	$1 - q_{xyz}^{ij} - r_{xyz}^{ij}$

Table 5.4: Amount of vectors $\mathbf{v}_1$, $\mathbf{v}_2$ and $\mathbf{v}_3$ to add to $\mathbf{v}_0$ to arrive at the position of intersection between the plane P_γ and the (ij)–th edge of X.

denoted the (ij)–th edge, has values of q_γ and r_γ equal to

$$q_\gamma^{ij} = \frac{q_\gamma(\mathbf{x}_j)p_\gamma(\mathbf{x}_i) - q_\gamma(\mathbf{x}_i)p_\gamma(\mathbf{x}_j)}{p_\gamma(\mathbf{x}_i) - p_\gamma(\mathbf{x}_j)}, \quad r_\gamma^{ij} = \frac{r_\gamma(\mathbf{x}_j)p_\gamma(\mathbf{x}_i) - r_\gamma(\mathbf{x}_i)p_\gamma(\mathbf{x}_j)}{p_\gamma(\mathbf{x}_i) - p_\gamma(\mathbf{x}_j)}. \tag{5.1}$$

The numerators of these values are listed in Table 5.3 in the form of determinants. The last row, for the plane P_{xyz}, has been simplified.

Equation (5.1) is the extension of equation (4.2). In 2D, only one coordinate is needed to determine the position of such an intersection. In 3D, the intersection lies within a 2D plane and so two coordinates are required. Likewise, table 5.3 is the extension of table 4.3.

This intersection lies on Y if and only if $q_\gamma^{ij} \geq 0$, $r_\gamma^{ij} \geq 0$ and $1 - q_\gamma^{ij} - r_\gamma^{ij} \geq 0$. Otherwise, the intersection does not fall on a face of Y and is not a corner of the polyhedron Z. Therefore, the sign of each of these must be found. This is trivial for the first two, while the last involves an additional calculation. Its numerator has been included in Table 5.3. Its denominator is the same as the others, $p_\gamma(\mathbf{x}_i) - p_\gamma(\mathbf{x}_j)$.

Each value in Table 5.3 appears twice. Thus, of twelve entries only six need to be calculated. This keeps calculations consistent and can improve efficiency.

If there is no intersection with one of the other planes then the signs of one of q_γ^{ij}, r_γ^{ij} or $1 - q_\gamma^{ij} - r_\gamma^{ij}$ has the same sign as $p_\gamma(\mathbf{x}_i)$ for this plane. For example, if $\text{sign}(p_y(\mathbf{x}_i)) = \text{sign}(p_y(\mathbf{x}_j)) = 1$ then the (ij)–th edge does not intersect the plane P_y and any intersection between this edge and another plane of Y must also be on the positive side of P_y. In this case, $q_x^{ij} \geq 0$, $r_z^{ij} \geq 0$ and $r_{xyz}^{ij} \geq 0$.

If an intersection is found to lie on Y then we require its coordinates in the original system. This position is equal to $\mathbf{v}_0 + a\mathbf{v}_1 + b\mathbf{v}_2 + c\mathbf{v}_3$, where the values of a, b and c depend on the face of Y the intersection lies. These values are listed in Table 5.4.

By Proposition 5.1 there are either 0, 3 or 4 intersections between X and the plane P_γ. Thus, these intersections, if they exist, form either a triangle or a quadrilateral, denoted G, that may or may not intersect the face of Y. By comparing the signs of q_γ^{ij}, r_γ^{ij} and $1 - q_\gamma^{ij} - r_\gamma^{ij}$ for different combinations of i and j, which have already been found, we can determine which, if any, edges of Y intersect the faces of X. Moreover, we can determine which faces of X these edges of Y intersect by taking the triple formed by the two combinations of i and j, noting that edges of G have one of these indices in common. For example, if $\text{sign}(q_\gamma^{ij}) \neq \text{sign}(q_\gamma^{ik})$ then the line $q_\gamma = 0$ intersects the plane formed by the i–th, j–th and k–th vertices of X.

5.3.3 Intersections between faces of X and edges of Y

Continuing the example from above, suppose there is an intersection between the (ijk)–th plane of X with the line $q_\gamma = 0$ for some γ. Suppose there is an edge of G between

γ,η	x	y	z	γ,η	x	y	z
$y,\ z$	$t_{y,z}$			$z,\ xyz$	$1-t_{z,xyz}$	$t_{z,xyz}$	
$x,\ z$		$t_{x,z}$		$y,\ xyz$	$t_{y,xyz}$		$1-t_{y,xyz}$
$x,\ y$			$t_{x,y}$	$x,\ xyz$		$1-t_{x,xyz}$	$t_{x,xyz}$

Table 5.5: Coordinates of points along edges of Y parametrized by $t_{\gamma,\eta}$.

its (ij)–th and (ik)–th vertices. Then the intersection along $q_\gamma = 0$ is

$$q_\gamma^0 = \frac{r_\gamma^{ik} q_\gamma^{ij} - r_\gamma^{ij} q_\gamma^{ik}}{q_\gamma^{ij} - q_\gamma^{ik}}.$$

As mentioned in Section 5.3.2, this is calculated only if $\operatorname{sign}(q_\gamma^{ij}) \neq \operatorname{sign}(q_\gamma^{ik})$.

Given that the edges of G are straightforward to determine, this procedure would find at most two intersections for each edge of a face of Y. However, each edge is shared by two faces of Y. For each intersection there are then two sets of calculations to produce it, with no guarantee that numerical error will keep them the same. Thus, we seek a single formula for each intersection that is independent of any particular face of Y.

Each edge of Y can be parametrized by a single value. We denote this parameter by $t_{\gamma,\eta}$ where γ and η indicate the planes of Y that intersect at the particular edge. This parameter is chosen such that the edge of Y lies between $t_{\gamma,\eta} = 0$ and $t_{\gamma,\eta} = 1$. The (x, y, z)–coordinates of points along these edges are listed in Table 5.5.

This table also provides the transformation of these points into the original coordinate system in the same manner as Table 5.4. Starting from position $\mathbf{v}_0$, add $\mathbf{v}_1$ times the x–coordinate, $\mathbf{v}_2$ times the y–coordinate and $\mathbf{v}_3$ times the z–coordinate. For example, the position of a point along the edge indexed by y, z is $\mathbf{v}_0 + t_{y,z}\mathbf{v}_1$.

Lemma 5.2. *The value of the parameter $t_{\gamma,\eta}$ at the intersection between the (ijk)–th plane of X and the line extending from the edge of Y indexed by γ, η is*

$$t_{y,z}^{ijk} = \frac{\begin{vmatrix} x_i & y_i & z_i \\ x_j & y_j & z_j \\ x_k & y_k & z_k \end{vmatrix}}{\begin{vmatrix} 1 & y_i & z_i \\ 1 & y_j & z_j \\ 1 & y_k & z_k \end{vmatrix}}, \qquad t_{z,xyz}^{ijk} = \frac{\begin{vmatrix} 1-x_i & y_i & z_i \\ 1-x_j & y_j & z_j \\ 1-x_k & y_k & z_k \end{vmatrix}}{\begin{vmatrix} 1 & x_i+y_i & z_i \\ 1 & x_j+y_j & z_j \\ 1 & x_k+y_k & z_k \end{vmatrix}},$$

$$t_{x,z}^{ijk} = \frac{\begin{vmatrix} x_i & y_i & z_i \\ x_j & y_j & z_j \\ x_k & y_k & z_k \end{vmatrix}}{\begin{vmatrix} x_i & 1 & z_i \\ x_j & 1 & z_j \\ x_k & 1 & z_k \end{vmatrix}}, \qquad t_{y,xyz}^{ijk} = \frac{\begin{vmatrix} x_i & y_i & 1-z_i \\ x_j & y_j & 1-z_j \\ x_k & y_k & 1-z_k \end{vmatrix}}{\begin{vmatrix} x_i+z_i & y_i & 1 \\ x_j+z_j & y_j & 1 \\ x_k+z_k & y_k & 1 \end{vmatrix}},$$

$$t^{ijk}_{x,y} = \frac{\begin{vmatrix} x_i & y_i & z_i \\ x_j & y_j & z_j \\ x_k & y_k & z_k \end{vmatrix}}{\begin{vmatrix} x_i & y_i & 1 \\ x_j & y_j & 1 \\ x_k & y_k & 1 \end{vmatrix}}, \qquad t^{ijk}_{x,xyz} = \frac{\begin{vmatrix} x_i & 1-y_i & z_i \\ x_j & 1-y_j & z_j \\ x_k & 1-y_k & z_k \end{vmatrix}}{\begin{vmatrix} x_i & 1 & y_i+z_i \\ x_j & 1 & y_j+z_j \\ x_k & 1 & y_k+z_k \end{vmatrix}}.$$

The value of $1 - t^{ijk}_{\gamma,\eta}$ *is*

$$1-t^{ijk}_{y,z} = \frac{\begin{vmatrix} 1-x_i & y_i & z_i \\ 1-x_j & y_j & z_j \\ 1-x_k & y_k & z_k \end{vmatrix}}{\begin{vmatrix} 1 & y_i & z_i \\ 1 & y_j & z_j \\ 1 & y_k & z_k \end{vmatrix}}, \qquad 1-t^{ijk}_{z,xyz} = -\frac{\begin{vmatrix} x_i & 1-y_i & z_i \\ x_j & 1-y_j & z_j \\ x_k & 1-y_k & z_k \end{vmatrix}}{\begin{vmatrix} 1 & x_i+y_i & z_i \\ 1 & x_j+y_j & z_j \\ 1 & x_k+y_k & z_k \end{vmatrix}},$$

$$1-t^{ijk}_{x,z} = \frac{\begin{vmatrix} x_i & 1-y_i & z_i \\ x_j & 1-y_j & z_j \\ x_k & 1-y_k & z_k \end{vmatrix}}{\begin{vmatrix} x_i & 1 & z_i \\ x_j & 1 & z_j \\ x_k & 1 & z_k \end{vmatrix}}, \qquad 1-t^{ijk}_{y,xyz} = -\frac{\begin{vmatrix} 1-x_i & y_i & z_i \\ 1-x_j & y_j & z_j \\ 1-x_k & y_k & z_k \end{vmatrix}}{\begin{vmatrix} x_i+z_i & y_i & 1 \\ x_j+z_j & y_j & 1 \\ x_k+z_k & y_k & 1 \end{vmatrix}},$$

$$1-t^{ijk}_{x,y} = \frac{\begin{vmatrix} x_i & y_i & 1-z_i \\ x_j & y_j & 1-z_j \\ x_k & y_k & 1-z_k \end{vmatrix}}{\begin{vmatrix} x_i & y_i & 1 \\ x_j & y_j & 1 \\ x_k & y_k & 1 \end{vmatrix}}, \qquad 1-t^{ijk}_{x,xyz} = -\frac{\begin{vmatrix} x_i & y_i & 1-z_i \\ x_j & y_j & 1-z_j \\ x_k & y_k & 1-z_k \end{vmatrix}}{\begin{vmatrix} x_i & 1 & y_i+z_i \\ x_j & 1 & y_j+z_j \\ x_k & 1 & y_k+z_k \end{vmatrix}}.$$

Proof Consider the (ijk)–th face of X. This face defines a plane, $ax + by + cz = d$. The intersection between this plane and the line y, z, where $y = z = 0$, is $(d/a, 0, 0)$. Likewise, for the lines x, z and x, y the intersections are $(0, d/b, 0)$ and $(0, 0, d/c)$, respectively.

Consider the intersection between this plane and the line x, xyz. This intersection is the solution to the linear system

$$\begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ a & b & c \end{bmatrix} \mathbf{x} = \begin{bmatrix} 0 \\ 1 \\ d \end{bmatrix}$$

which is $(0, -(c-d)/(b-c), (b-d)/(b-c))$. The intersections between the plane and the lines y, xyz and z, xyz can be found in the same manner.

The values of a, b and c can be found by solving the linear system

$$\begin{bmatrix} x_i & y_i & z_i \\ x_j & y_j & z_j \\ x_k & y_k & z_k \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = d \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}.$$

Cramer's rule gives the solution as

$$\frac{a}{d} = \frac{\begin{vmatrix} 1 & y_i & z_i \\ 1 & y_j & z_j \\ 1 & y_k & z_k \end{vmatrix}}{\begin{vmatrix} x_i & y_i & z_i \\ x_j & y_j & z_j \\ x_k & y_k & z_k \end{vmatrix}}, \quad \frac{b}{d} = \frac{\begin{vmatrix} x_i & 1 & z_i \\ x_j & 1 & z_j \\ x_k & 1 & z_k \end{vmatrix}}{\begin{vmatrix} x_i & y_i & z_i \\ x_j & y_j & z_j \\ x_k & y_k & z_k \end{vmatrix}}, \quad \frac{c}{d} = \frac{\begin{vmatrix} x_i & y_i & 1 \\ x_j & y_j & 1 \\ x_k & y_k & 1 \end{vmatrix}}{\begin{vmatrix} x_i & y_i & z_i \\ x_j & y_j & z_j \\ x_k & y_k & z_k \end{vmatrix}}.$$

The values of $t^{ijk}_{y,z}$, $t^{ijk}_{x,z}$ and $t^{ijk}_{x,y}$ are then the inverses of these fractions. The values of $1-t^{ijk}_{y,z}$, $1-t^{ijk}_{x,z}$ and $1-t^{ijk}_{x,y}$ can be simplified using known properties of the determinant.

The value of $t^{ijk}_{x,xyz}$ is

$$t^{ijk}_{x,xyz} = \frac{b-d}{b-c} = \frac{\frac{b}{d}-1}{\frac{b}{d}-\frac{c}{d}} = \frac{\begin{vmatrix} x_i & 1 & z_i \\ x_j & 1 & z_j \\ x_k & 1 & z_k \end{vmatrix} - \begin{vmatrix} x_i & y_i & z_i \\ x_j & y_j & z_j \\ x_k & y_k & z_k \end{vmatrix}}{\begin{vmatrix} x_i & 1 & z_i \\ x_j & 1 & z_j \\ x_k & 1 & z_k \end{vmatrix} - \begin{vmatrix} x_i & y_i & 1 \\ x_j & y_j & 1 \\ x_k & y_k & 1 \end{vmatrix}}$$

$$= \frac{\begin{vmatrix} x_i & 1-y_i & z_i \\ x_j & 1-y_j & z_j \\ x_k & 1-y_k & z_k \end{vmatrix}}{\begin{vmatrix} x_i & 1 & z_i \\ x_j & 1 & z_j \\ x_k & 1 & z_k \end{vmatrix} + \begin{vmatrix} x_i & 1 & y_i \\ x_j & 1 & y_j \\ x_k & 1 & y_k \end{vmatrix}} = \frac{\begin{vmatrix} x_i & 1-y_i & z_i \\ x_j & 1-y_j & z_j \\ x_k & 1-y_k & z_k \end{vmatrix}}{\begin{vmatrix} x_i & 1 & y_i+z_i \\ x_j & 1 & y_j+z_j \\ x_k & 1 & y_k+z_k \end{vmatrix}}.$$

The value of $1 - t^{ijk}_{x,xyz}$ has already been shown to be $-(c-d)/(b-c)$ and a similar expression as above can be found by following the same steps. The remaining values of $t^{ijk}_{\gamma,\eta}$ and $1 - t^{ijk}_{\gamma,\eta}$ are also found in this manner. ■

If $t^{ijk}_{\gamma,\eta}$ is between 0 and 1 then the corresponding intersection lies on the corresponding edge of Y. Note that the values of $1 - t^{ijk}_{\gamma,\eta}$ do not need to be calculated explicitly as only their signs are important.

In the case where $\text{sign}(r^{ij}_\gamma) = \text{sign}(r^{ik}_\gamma)$ the sign of $t^{ijk}_{\gamma,\eta}$ can be determined without additional calculations. Since the line connecting the (ij)–th intersection to the (ik)–th intersection lies entirely on one side of the line $r_\gamma = 0$ the sign of $t^{ijk}_{\gamma,\eta}$ must be equal to both $\text{sign}(r^{ij}_\gamma)$ and $\text{sign}(r^{ik}_\gamma)$. For some edges r_γ is replaced by q_γ or $1 - q_\gamma - r_\gamma$ and $t_{\gamma,\eta}$ by $1 - t_{\gamma,\eta}$. In this way, the sign of $t^{ijk}_{\gamma,\eta}$ needs to be independently calculated only when the (ijk)–th plane of X intersects three lines extending from the edges of Y.

Each numerator appears in Lemma 5.2 three times. Each denominator appears twice. The numerators are connected by the vertices of Y: All edges extending from a given vertex of Y share the numerator of either $t^{ijk}_{\gamma,\eta}$ or $1 - t^{ijk}_{\gamma,\eta}$. The denominators are specific to each edge. These denominators have common signs with the numerators of Table 5.3.

Lemma 5.3. *Suppose the (ijk)–th face of X intersects the line γ, η of Y. Suppose*

$\operatorname{sign}(p_\gamma(\mathbf{x}_i)) \neq \operatorname{sign}(p_\gamma(\mathbf{x}_j)) = \operatorname{sign}(p_\gamma(\mathbf{x}_k))$. *If the line γ, η coincides with $q_\gamma = 0$ then*

$$\operatorname{sign}\left(\begin{vmatrix} 1 & p_\gamma(\mathbf{x}_i) & q_\gamma(\mathbf{x}_i) \\ 1 & p_\gamma(\mathbf{x}_j) & q_\gamma(\mathbf{x}_j) \\ 1 & p_\gamma(\mathbf{x}_k) & q_\gamma(\mathbf{x}_k) \end{vmatrix}\right) = \operatorname{sign}\left(\begin{vmatrix} p_\gamma(\mathbf{x}_i) & q_\gamma(\mathbf{x}_i) \\ p_\gamma(\mathbf{x}_j) & q_\gamma(\mathbf{x}_j) \end{vmatrix}\right).$$

If the line coincides with $q_\gamma + r_\gamma = 1$ then

$$\operatorname{sign}\left(\begin{vmatrix} 1 & q_\gamma(\mathbf{x}_i) + r_\gamma(\mathbf{x}_i) & p_\gamma(\mathbf{x}_i) \\ 1 & q_\gamma(\mathbf{x}_j) + r_\gamma(\mathbf{x}_j) & p_\gamma(\mathbf{x}_j) \\ 1 & q_\gamma(\mathbf{x}_k) + r_\gamma(\mathbf{x}_k) & p_\gamma(\mathbf{x}_k) \end{vmatrix}\right) = \operatorname{sign}\left(\begin{vmatrix} p_\gamma(\mathbf{x}_i) & 1 - q_\gamma(\mathbf{x}_i) - r_\gamma(\mathbf{x}_i) \\ p_\gamma(\mathbf{x}_j) & 1 - q_\gamma(\mathbf{x}_j) - r_\gamma(\mathbf{x}_j) \end{vmatrix}\right).$$

Proof The intersection along $q_\gamma = 0$ has already been given and involves division by $q_\gamma^{ij} - q_\gamma^{ik}$. The value of this denominator is

$$\begin{aligned} q_\gamma^{ij} - q_\gamma^{ik} =& \frac{\begin{vmatrix} p_\gamma(\mathbf{x}_i) & q_\gamma(\mathbf{x}_i) \\ p_\gamma(\mathbf{x}_j) & q_\gamma(\mathbf{x}_j) \end{vmatrix}}{p_\gamma(\mathbf{x}_i) - p_\gamma(\mathbf{x}_j)} - \frac{\begin{vmatrix} p_\gamma(\mathbf{x}_i) & q_\gamma(\mathbf{x}_i) \\ p_\gamma(\mathbf{x}_k) & q_\gamma(\mathbf{x}_k) \end{vmatrix}}{p_\gamma(\mathbf{x}_i) - p_\gamma(\mathbf{x}_k)} \\ =& \frac{p_\gamma(\mathbf{x}_i)\begin{vmatrix} 0 & p_\gamma(\mathbf{x}_i) & q_\gamma(\mathbf{x}_i) \\ 1 & p_\gamma(\mathbf{x}_j) & q_\gamma(\mathbf{x}_j) \\ 1 & p_\gamma(\mathbf{x}_k) & q_\gamma(\mathbf{x}_k) \end{vmatrix} - \begin{vmatrix} 0 & p_\gamma(\mathbf{x}_i) & q_\gamma(\mathbf{x}_i) \\ p_\gamma(\mathbf{x}_j) & p_\gamma(\mathbf{x}_j) & q_\gamma(\mathbf{x}_j) \\ p_\gamma(\mathbf{x}_k) & p_\gamma(\mathbf{x}_k) & q_\gamma(\mathbf{x}_k) \end{vmatrix}}{(p_\gamma(\mathbf{x}_i) - p_\gamma(\mathbf{x}_j))(p_\gamma(\mathbf{x}_i) - p_\gamma(\mathbf{x}_k))} \\ =& \frac{p_\gamma(\mathbf{x}_i)\begin{vmatrix} 0 & p_\gamma(\mathbf{x}_i) & q_\gamma(\mathbf{x}_i) \\ 1 & p_\gamma(\mathbf{x}_j) & q_\gamma(\mathbf{x}_j) \\ 1 & p_\gamma(\mathbf{x}_k) & q_\gamma(\mathbf{x}_k) \end{vmatrix} - \begin{vmatrix} -p_\gamma(\mathbf{x}_i) & p_\gamma(\mathbf{x}_i) & q_\gamma(\mathbf{x}_i) \\ 0 & p_\gamma(\mathbf{x}_j) & q_\gamma(\mathbf{x}_j) \\ 0 & p_\gamma(\mathbf{x}_k) & q_\gamma(\mathbf{x}_k) \end{vmatrix}}{(p_\gamma(\mathbf{x}_i) - p_\gamma(\mathbf{x}_j))(p_\gamma(\mathbf{x}_i) - p_\gamma(\mathbf{x}_k))} \\ =& \frac{p_\gamma(\mathbf{x}_i)\begin{vmatrix} 1 & p_\gamma(\mathbf{x}_i) & q_\gamma(\mathbf{x}_i) \\ 1 & p_\gamma(\mathbf{x}_j) & q_\gamma(\mathbf{x}_j) \\ 1 & p_\gamma(\mathbf{x}_k) & q_\gamma(\mathbf{x}_k) \end{vmatrix}}{(p_\gamma(\mathbf{x}_i) - p_\gamma(\mathbf{x}_j))\,(p_\gamma(\mathbf{x}_i) - p_\gamma(\mathbf{x}_k))}. \end{aligned}$$

Since this intersection exists only if $\operatorname{sign}(q_\gamma^{ij}) \neq \operatorname{sign}(q_\gamma^{ik})$ this denominator has the same sign as q_γ^{ij}, which has the sign of

$$\begin{vmatrix} p_\gamma(\mathbf{x}_i) & q_\gamma(\mathbf{x}_i) \\ p_\gamma(\mathbf{x}_j) & q_\gamma(\mathbf{x}_j) \end{vmatrix} p_\gamma(\mathbf{x}_i).$$

By comparing these signs it is clear that

$$\operatorname{sign}\left(\begin{vmatrix} 1 & p_\gamma(\mathbf{x}_i) & q_\gamma(\mathbf{x}_i) \\ 1 & p_\gamma(\mathbf{x}_j) & q_\gamma(\mathbf{x}_j) \\ 1 & p_\gamma(\mathbf{x}_k) & q_\gamma(\mathbf{x}_k) \end{vmatrix}\right) = \operatorname{sign}\left(\begin{vmatrix} p_\gamma(\mathbf{x}_i) & q_\gamma(\mathbf{x}_i) \\ p_\gamma(\mathbf{x}_j) & q_\gamma(\mathbf{x}_j) \end{vmatrix}\right).$$

The intersection along $q_\gamma + r_\gamma = 1$ involves division by $(1 - q_\gamma^{ij} - r_\gamma^{ij}) - (1 - q_\gamma^{ik} - r_\gamma^{ik})$. Again, this only exists if these two terms differ in sign and this denominator has the same sign as $1 - q_\gamma^{ij} - r_\gamma^{ij}$, which is

$$\operatorname{sign}\left(\begin{vmatrix} p_\gamma(\mathbf{x}_i) & 1 - q_\gamma(\mathbf{x}_i) - r_\gamma(\mathbf{x}_i) \\ p_\gamma(\mathbf{x}_j) & 1 - q_\gamma(\mathbf{x}_j) - r_\gamma(\mathbf{x}_j) \end{vmatrix} p_\gamma(\mathbf{x}_i)\right).$$

Type of corner	Shorthand
Vertex of X inside Y	X–in–Y
Intersection, edge of X and face of Y	X–with–Y
Intersection, edge of Y and face of X	Y–with–X
Vertex of Y inside X	Y–in–X

Table 5.6: Hierarchy of corners of the polyhedron of intersection.

The value of the denominator can be found by replacing q_γ by $1 - q_\gamma - r_\gamma$ in the formulas above. To arrive at the form in the statement of the lemma one must simplify the determinant

$$\begin{aligned}\begin{vmatrix}1 & p_\gamma(\mathbf{x}_i) & 1-q_\gamma(\mathbf{x}_i)-r_\gamma(\mathbf{x}_i)\\ 1 & p_\gamma(\mathbf{x}_j) & 1-q_\gamma(\mathbf{x}_j)-r_\gamma(\mathbf{x}_j)\\ 1 & p_\gamma(\mathbf{x}_k) & 1-q_\gamma(\mathbf{x}_k)-r_\gamma(\mathbf{x}_k)\end{vmatrix} &= \begin{vmatrix}1 & p_\gamma(\mathbf{x}_i) & 1\\ 1 & p_\gamma(\mathbf{x}_j) & 1\\ 1 & p_\gamma(\mathbf{x}_k) & 1\end{vmatrix} - \begin{vmatrix}1 & p_\gamma(\mathbf{x}_i) & q_\gamma(\mathbf{x}_i)+r_\gamma(\mathbf{x}_i)\\ 1 & p_\gamma(\mathbf{x}_j) & q_\gamma(\mathbf{x}_j)+r_\gamma(\mathbf{x}_j)\\ 1 & p_\gamma(\mathbf{x}_k) & q_\gamma(\mathbf{x}_k)+r_\gamma(\mathbf{x}_k)\end{vmatrix}\\ &= \begin{vmatrix}1 & q_\gamma(\mathbf{x}_i)+r_\gamma(\mathbf{x}_i) & p_\gamma(\mathbf{x}_i)\\ 1 & q_\gamma(\mathbf{x}_j)+r_\gamma(\mathbf{x}_j) & p_\gamma(\mathbf{x}_j)\\ 1 & q_\gamma(\mathbf{x}_k)+r_\gamma(\mathbf{x}_k) & p_\gamma(\mathbf{x}_k)\end{vmatrix}.\end{aligned}$$

Comparison with the known sign of this denominator results in the statement of the lemma. ■

This connects the denominators of Lemma 5.2 with the numerators of Table 5.3. Depending on the indices and the particular q_γ and p_γ some row and column swapping may be necessary. Each swap incurs a sign change. Note also that q_γ will be replaced by r_γ in some instances. Since each edge of Y is shared by two of its faces there is no need to use $\gamma = xyz$ to connect these signs.

5.3.4 Vertices of Y that lie inside X

Each edge of Y has two vertices attached to it. These are located at $t_{\gamma,\eta} = 0$ and $t_{\gamma,\eta} = 1$. Either two or zero faces of X intersect the line extending from this edge, resulting in $t^{ijk}_{\gamma,\eta}$ and $t^{ijl}_{\gamma,\eta}$. If the signs of these values are different then the vertex at $t_{\gamma,\eta} = 0$ must lie inside X. The same is true of $1 - t^{ijk}_{\gamma,\eta}$, $1 - t^{ijl}_{\gamma,\eta}$ and the vertex at $t_{\gamma,\eta} = 1$.

Each vertex of Y has three edges extending from it. This test can therefore occur up to three times. If PANG2-3D is consistent it needs to only occur once. The remaining edges would then agree on the results. Using the edges of the plane P_{xyz} for these tests removes the need to test the signs of $1-t^{ijk}_{\gamma,\eta}$ as each of these edges has a separate vertex of Y at $t_{\gamma,xyz} = 0$. The final vertex of Y, at the origin, can use any of the remaining edges, as they all have this vertex at $t_{\gamma,\eta} = 0$.

5.4 Algorithm

There are four types of points to find to construct the polyhedron of intersection. They are presented in Table 5.6.

Algorithm 5.1 (PANG2-3D)**. Step 1: Change of coordinates.** *As described in Section 5.2, V is transformed into Y and U into X.*

Edge	$\mathbf{v}_1$	$\mathbf{v}_2$	$\mathbf{v}_3$
$y = 0, z = 0$	t		
$x = 0, z = 0$		t	
$x = 0, y = 0$			t
$z = 0, x + y + z = 1$	$1 - t$	t	
$y = 0, x + y + z = 1$	t		$1 - t$
$x = 0, x + y + z = 1$		$1 - t$	t

Table 5.7: Position of the intersection between an edge of Y and a face of X in original coordinates as functions of its t–coordinate.

Step 2: Select plane P_γ of Y. *Calculate $p_\gamma(\mathbf{x}_i)$, $q_\gamma(\mathbf{x}_i)$ and $r_\gamma(\mathbf{x}_i)$ for all vertices of X, $i = 1, \dots, 4$.*

Step 2(i): Intersections between P_γ and edges of X. *Test if* $\operatorname{sign}(p_\gamma(\mathbf{x}_i)) \neq \operatorname{sign}(p_\gamma(\mathbf{x}_j))$. *If so, calculate the intersection between $p_\gamma = 0$ and the edge connecting the i–th and j–th vertices of X, found using the coordinates q^{ij}_γ and r^{ij}_γ. If q^{ij}_γ, r^{ij}_γ and $1 - q^{ij}_\gamma - r^{ij}_\gamma$ are all non-negative then the intersection is a corner of the polyhedron of intersection. Repeat for all pairs of i and j.*

Step 2(ii): Intersection between G and edges of Y. *Select an edge of Y that lies in P_γ. Without loss of generality, suppose this aligns with $q_\gamma = p_\eta = 0$. Test if* $\operatorname{sign}(q^{ij}_\gamma) \neq \operatorname{sign}(q^{ik}_\gamma)$ *If so, calculate the intersection $t^{ijk}_{\gamma,\eta}$ between G and this edge of Y. If $t^{ijk}_{\gamma,\eta} \in [0, 1]$ then this intersection is a corner of the polyhedron of intersection. Repeat for all pairs of ij and ik.*

Step 2(ii.a): Intersections between edges of Y and faces of X. *Test if* $\operatorname{sign}(t^{ijk}_{\gamma,\eta}) \neq \operatorname{sign}(t^{ijl}_{\gamma,\eta})$. *If so, the vertex of Y at $t_{\gamma,\eta} = 0$ lies in X. Likewise, if* $\operatorname{sign}(1 - t^{ijk}_{\gamma,\eta}) \neq \operatorname{sign}(1 - t^{ijl}_{\gamma,\eta})$ *then the vertex of Y at $t_{\gamma,\eta} = 1$ lies in X.*

Repeat step 2(ii) for each edge of Y that lies in P_γ.

Repeat step 2 for each face of Y. It is not necessary to repeat step 2(ii) for edges considered in previous instances of step 2, nor step 2(ii.a) for previously considered vertices.

Step 3: Vertices of X in Y. *Compare all values of $p_\gamma(\mathbf{x}_i)$ for the i–th vertex of X. If* $\operatorname{sign}(p_\gamma(\mathbf{x}_i)) = 1$ *for all γ then the i–th vertex lies inside Y.*

Step 4: Undo change of coordinates. *Transform the intersections between edges of X and faces of Y (step 2(i)) and those between edges of Y and faces of X (step 2(ii)) into the original coordinates. The transformation from (p, q, r)–coordinates has been shown in Table 5.4. The transformation from t–coordinate is presented in Table 5.7. Take the numbers listed in the tables, multiply by the respective vectors $\mathbf{v}_i$, sum the results and add $\mathbf{v}_0$ for the positions in original coordinates.*

Note that step 2(i) may be completed after step 2(i) has been completed for all planes P_γ. Likewise, step 2(ii.a) may be completed after all instances of step 2(ii). This means the algorithm can be parallelized by having these steps run by separate threads. One could also have each plane P_γ run by separate threads, though information about which edges and vertices have already been considered would need to be passed between these threads.

5.5 Consistency errors

If PANG2-3D is to be robust, an error on the order of machine epsilon can only cause a change in the volume of the polyhedron of intersection on the same order of magnitude. The polyhedron is defined by its corners. The types of corners are listed in Table 5.6. There are two types of corners arising from intersections, which may have error in their position, and two types of corners arising from vertices of the tetrahedra, which may have error in their inclusion in the polyhedron.

5.5.1 X–in–Y errors

The vertices of the tetrahedron Y are fixed. The vertices of the tetrahedron X are determined by an affine transformation. This transformation may introduce some error in their position but as only the original positions of the vertices are used to construct the polyhedron this only affects the determination of its inclusion as an X–in–Y corner and the position of any intersections calculated based on these vertices.

Should a vertex of X cross a plane of Y it is crucial that the number of X–with–Y points in the plane changes accordingly. If this were not the case, the resulting polyhedron of intersection may not represent a realistic intersection. For example, if the exact intersection has four corners, three X–with–Y points and an X–in–Y vertex, the movement of the vertex over the plane will result in a 2D intersection of 3D objects.

Lemma 5.4. *Let $\mathbf{x}_i$ be the position of the i–th vertex and $\tilde{\mathbf{x}}_i$ be its position as calculated by PANG2-3D. Suppose the i–th vertex of X lies outside Y but the algorithm determines it to lie inside Y. The line segment between $\mathbf{x}_i$ and $\tilde{\mathbf{x}}_i$ necessarily intersects at least one plane P_γ. The line segment between $\tilde{\mathbf{x}}_i$ and $\mathbf{x}_j$ intersects P_γ if and only if the (ij)–th edge of X does not.*

Proof By assumption, $\text{sign}(p_\gamma(\mathbf{x}_i)) = 0$ and $\text{sign}\,(\tilde{\mathbf{x}}_i) = 1$. Then either

$$\text{sign}(p_\gamma(\mathbf{x}_j)) \neq \text{sign}(p_\gamma(\mathbf{x}_i))$$

or

$$\text{sign}(p_\gamma(\mathbf{x}_j)) \neq \text{sign}\left(p_\gamma\left(\tilde{\mathbf{x}}_i\right)\right).$$

The former indicates the (ij)–th edge of X intersects P_γ and the line segment between $\tilde{\mathbf{x}}_i$ and $\mathbf{x}_j$ does not. The latter indicates the reverse. ■

The number of intersections with the plane after the error depends on the number of vertices of X on each side of the plane. We consider the five possible arrangements of the vertices with respect to the plane and indicate the change in the number of intersections found when an error causes transition from one to another, see Figure 5.1. These transitions correspond to moving between the rows of Table 5.1.

Errors in X–in–Y points then cause errors in the number of X–with–Y points. However, these errors provide consistency between these types of points, so that the ultimate configuration remains an intersection between tetrahedra.

5.5.2 X–with–Y errors

The corners denoted as X–with–Y are intersections between edges of X and faces of Y. An X–with–Y corner on the polyhedron Z is an intersection between an edge of X

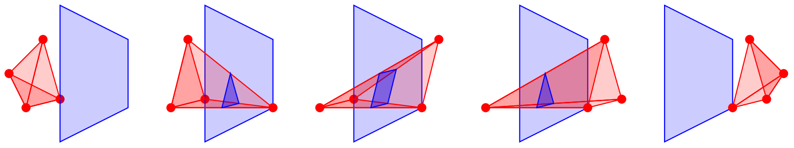


Figure 5.1: The five possible configurations of X with respect to a plane of Y. An X–in–Y error in a single vertex of X may cause a shift from one of these to either neighbour.

and a plane P_γ that lies between the lines $q_\gamma = 0$, $r_\gamma = 0$ and $q_\gamma + r_\gamma = 1$. A consistency error involving one of these corners then places one of these intersections on the wrong side of one of these lines. To maintain consistency this must cause commensurate errors on other planes.

An X–with–Y corner is determined by testing if the values of q_γ^{ij}, r_γ^{ij} and $1 - q_\gamma^{ij} - r_\gamma^{ij}$ are positive. The numerators of these values are found in Table 5.3. Change in the sign of one of these values is a change in the sign of the respective numerator. As this numerator is shared with another plane for the same edge of X this causes a second error. Should an error occur with an X–with–Y corner it is important that the number of Y–with–X corners remains consistent.

Lemma 5.5. *Suppose there is an error in the sign of q_γ^{ij} independently of any errors in $\text{sign}(p_\gamma(\mathbf{x}_i))$ and $\text{sign}(p_\gamma(\mathbf{x}_j))$. Then a value of $t_{\gamma,\eta}^{ijk}$ is calculated if and only if the (ijk)–th plane of X does not intersect the γ,η edge of Y.*

Proof If the sign of q_γ^{ij} is in error then so is that of r_η^{ij}. If no value of r_η^{ij} is calculated then $p_\eta(\mathbf{x}_i)$ and $p_\eta(\mathbf{x}_j)$ have the same sign. The (ij)–th edge of X then does not intersect the plane P_η and the sign of q_γ^{ij} cannot be in error.

By Lemma 5.4 there are two or three other intersections with the plane P_γ. The same is true for the plane P_η. One of these intersections has indices k and either i or j. Both q_γ^{ik} and q_γ^{jk} cannot be calculated as $\text{sign}(p_\gamma(\mathbf{x}_k))$ must equal either $\text{sign}(p_\gamma(\mathbf{x}_i))$ or $\text{sign}(p_\gamma(\mathbf{x}_j))$.

Without loss of generality we suppose there is a value of q_γ^{ik} and a value of r_η^{jk}. Before the error, both pairs of $(q_\gamma^{ij}, q_\gamma^{ik})$ and $(r_\eta^{ij}, r_\eta^{jk})$ either agreed or disagreed on their signs. After the error, the signs of both q_γ^{ij} and r_η^{ij} have flipped. Both pairs now have the opposite relationship between their signs. That is, if the pairs agreed on their signs before the error they now disagree after the error, and vice versa. Since agreement indicates no intersection between the γ, η edge of Y and the (ijk)–th plane of X and disagreement the reverse this concludes the proof. ■

If $\gamma = xyz$ then r_η^{ij} must be replaced by $1 - q_x^{ij} - r_x^{ij}$ in the statement of Lemma 5.5. The lemma is also true when replacing q_γ^{ij} with r_γ^{ij} or $1 - q_\gamma^{ij} - r_\gamma^{ij}$, making commenserate changes to r_η^{ij} where applicable. The end result ensures that an appropriate number of Y–with–X points are calculated based on the distribution of X–with–Y points, themselves ensured by the X vertices and their values of $\text{sign}(p_\gamma(\mathbf{x}_i))$.

Given that each edge of X has two faces attached to it the effects of Lemma 5.5 occur twice. If neither face intersects the line γ, η then after the error both do. If one face intersects the line and the other does not the intersection moves from one face to the other. If both faces intersect the line then after the error neither do. Figure 5.2 shows these three possible results of an X–with–Y error.

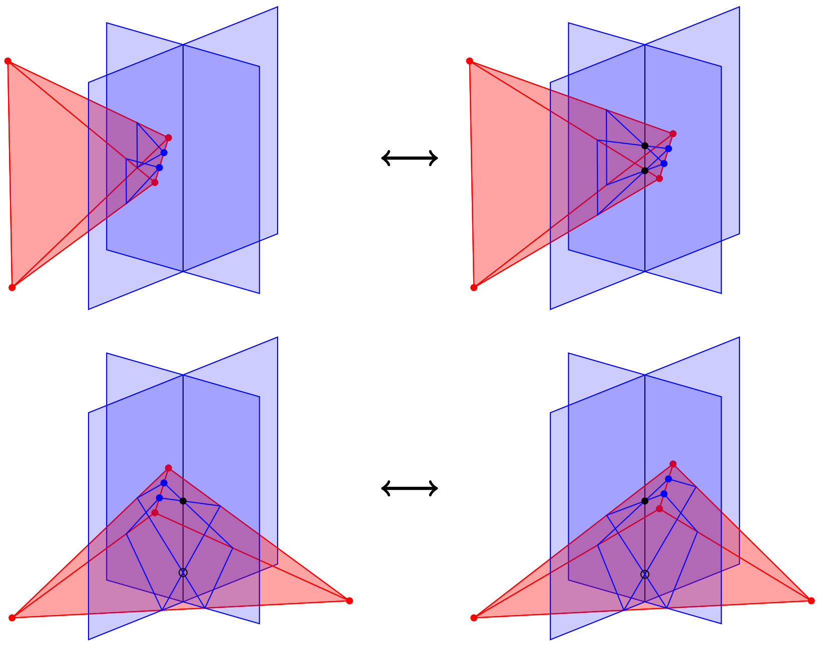


Figure 5.2: Possible X–with–Y errors. Blue planes are two planes of Y, P_γ and P_η. The blue lines trace out the intersections between X and the given plane. Blue dots represent X–with–Y points affected by error, black dots Y–with–X points altered by this error and black circles Y–with–X points unaffected.

This indicates that an X–with–Y error can only create two, destroy two or move one Y–with–X point(s). In a tetrahedral intersection Y–with–X points must come in pairs, as each line must enter then exit a convex object. Thus, X–with–Y errors maintain this parity.

In the absence of an intersection between this edge of X and the second plane of Y the sign of q_γ^{ij}, r_γ^{ij} or $1 - q_\gamma^{ij} - r_\gamma^{ij}$ is determined by the signs of $p_\eta(\mathbf{x}_i)$ and $p_\eta(\mathbf{x}_j)$, where P_η is the second plane sharing the numerator of the relevant value.

5.5.3 Y–with–X errors and Y–in–X errors

A Y–with–X corner is an intersection between an edge of Y and a face of X. In our algorithm it is represented by $0 \leq t_{\gamma,\eta}^{ijk} \leq 1$. A consistency error for this type of corner is then an error in the sign of $t_{\gamma,\eta}^{ijk}$ or $1 - t_{\gamma,\eta}^{ijk}$.

To maintain consistency an error in $\text{sign}(t_{\gamma,\eta}^{ijk})$ must affect the number of vertices of Y that are found inside X. That is, a Y–with–X error must cause a Y–in–X error.

Lemma 5.6. *Suppose there is an error in the sign of $t_{\gamma,\eta}^{ijk}$. Then the vertex of Y at $t_{\gamma,\eta} = 0$ is determined to lie within X if and only if it does not.*

Proof If the sign of $t_{\gamma,\eta}^{ijk}$ is in error then so is that of $t_{\gamma,\nu}^{ijk}$ and $t_{\nu,\eta}^{ijk}$. If one or more of these values does not exist then there can be no error in the sign of $t_{\gamma,\eta}^{ijk}$. For example, if $t_{\gamma,\nu}^{ijk}$ is not calculated then $\text{sign}(r_\gamma^{ij}) = \text{sign}(r_\gamma^{ik}) = \text{sign}(t_{\gamma,\eta}^{ijk})$. Some parameters and indices may need to be changed for this example to apply.

One of the other faces of X intersects the line γ, η of Y. This is true also for the lines γ, ν and ν, η. Denote the intersections between these lines and the other faces of X as $t_{\gamma,\eta}^*$, $t_{\gamma,\nu}^*$ and $t_{\nu,\eta}^*$.

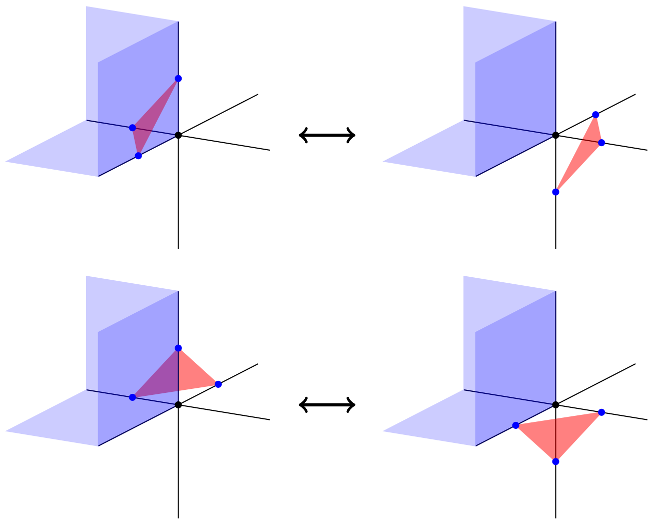


Figure 5.3: The four configurations of a face of X and a vertex of Y. A Y–with–X error transforms these configurations. The double-headed arrows indicate the directions of the transformations.

Before the error in the sign of $t^{ijk}_{\gamma,\eta}$ the three pairs of (t^{ijk}, t^*) shared a relation between $\text{sign}(t^{ijk})$ and $\text{sign}(t^*)$. That is, either $\text{sign}(t^{ijk}) = \text{sign}(t^*)$ for all three lines or $\text{sign}(t^{ijk}) \neq \text{sign}(t^*)$. After the error this relation switched, changing from agreement in sign to disagreement or the reverse. In the case of the former, the vertex of Y at the intersection of these three lines does not lie in X but is determined to do so by the algorithm. In the latter, the vertex lies in X but the algorithm does not place it there. ■

There are four possible configurations of the three Y–with–X points surrounding a vertex of Y. This error transforms these configurations between each other, see Figure 5.3.

5.5.4 Conclusions on consistency

Each of the errors described in this section can be represented by a change in the position of one or more of the vertices of X. This change is geometric. Therefore, the result remains an intersection between two tetrahedra.

The X–in–Y errors are already represented as the movement of vertices of X. Lemma 5.4 ensures the correct number of intersections with each of the planes P_γ are calculated.

For an X–with–Y error, the two intersections that change position lie along the same edge. This error can then be represented by the movement of the two vertices of X connected by this edge. Lemma 5.5 gives the three possible changes that result from this shift of an edge of X.

Finally, the three intersections altered by Y–with–X errors fall on the same face of X. The error is therefore equivalent to the shift of the three vertices of X connected by this face. Lemma 5.6 provides the four changes in configuration arising from the movement of a face of X.

Thus, the errors are equivalent to shifts in vertices, edges and faces of X. After these errors, X is still a tetrahedron and the result of PANG2-3D an intersection between two tetrahedra.

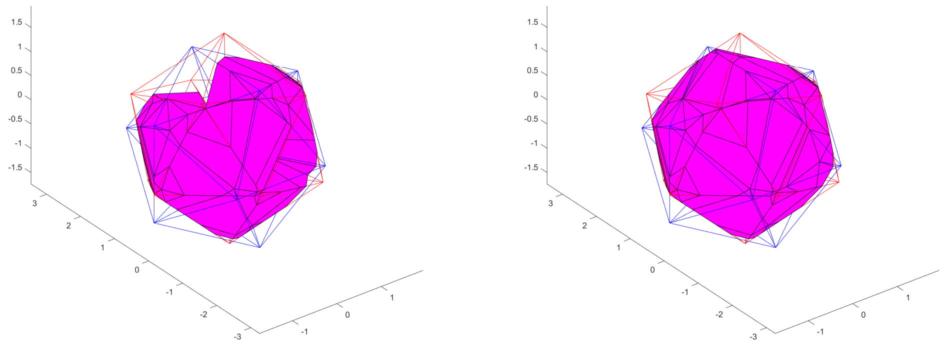


Figure 5.4: PANG (left) and PANG2-3D (right) applied to two intersecting icosahedral meshes.

5.6 Examples

We apply PANG2-3D to two examples to show its effectiveness. We compare it against its predecessor, PANG.

5.6.1 Intersecting icosahedra

In Chapter 4 we considered the intersection between two triangular meshes, both with 20 triangles arranged in a wheel, one of which was slightly rotated with its centre perturbed. We extend this example to 3D by considering two icosahedra, with each face representing a tetrahedron. All tetrahedra on the same icosahedron share the centre of the icosahedron as its fourth vertex. As before, one is rotated in two senses relative to the other, with its centre translated by a small ϵ. We have referred to this example as the Hecht example, as it was originally motiviated by discussions with Frédéric Hecht, see Section 4.9 for the 2D version.

Figure 5.4 shows the results of applying the two algorithms to the Hecht example. PANG has clearly missed several intersections. PANG2-3D finds the intersection exactly.

5.6.2 Accuracy and computation time

The second example in Chapter 4 was that of two intersecting triangles with a known intersection that changed with a given angle. To extend this to 3D we add one vertex to each of the two triangles. The vertices of the triangles V and U are found in Table 4.5. For both vertices let the x–coordinates be the first row of this table, the z–coordiantes the second, and the y–coordinates all set to zero. To V add the vertex $(0, 1, 0)$, and to U add $(0, 0.5, 0)$. The resulting coordinates are presented in Table 5.8. These tetrahedra may be seen for $\alpha = \pi/3$ and $2\pi/3$ in Figure 5.5

The intersection between U and V is effectively a pyramid with base equal to the intersection of the example in 2D. The area of the base may then be found in equation

V				$2 * U$			
0	0	0	$\cos(\alpha)$	$\cos(\alpha/2)$	$\cos(\alpha/2+2\pi/3)$	$\cos(\alpha/2+4\pi/3)$	0
0	1	0	0	0	0	0	1
0	0	1	$\sin(\alpha)$	$\sin(\alpha/2)$	$\sin(\alpha/2+2\pi/3)$	$\sin(\alpha/2+4\pi/3)$	0

Table 5.8: Coordinates of the clipping and subject tetrahedra. The coordinates of U must be multiplied by 0.5.

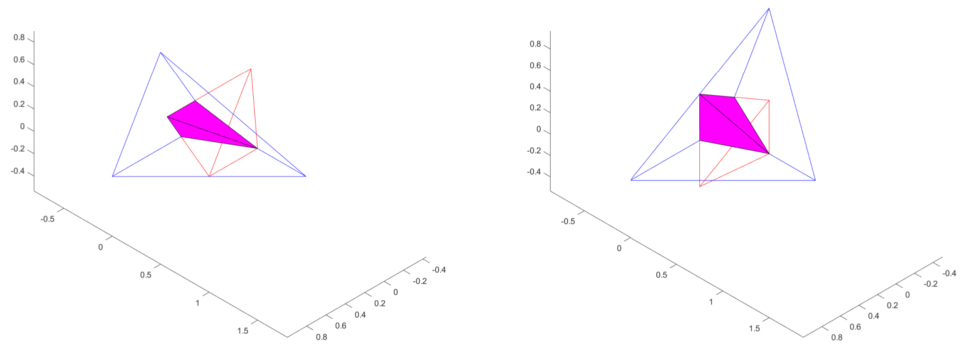


Figure 5.5: Intersecting tetrahedra for $\alpha = \pi/3$ (left) and $\alpha = 2\pi/3$ (right).

(4.4), and will be represented here as A_0. The pyramid has a height of 0.5, and so using the formula for the volume of a pyramid the intersection has volume $A_0/6$.

For large enough α, part of this pyramid again sticks out the other side of V. This portion is again a pyramid with base equal to that portion that stuck out in 2D. Let the area of the base be A_b, see equation (4.3). The height is the intersection of the lines $1/2 - y$ and $\cos(\alpha/2)(1-y)$, the heights of the $x = 0$ slices U and V, respectively. The height is then

$$\frac{\cos(\alpha/2) - 1/2}{\cos(\alpha/2) - 1}.$$

The total volume is then

$$V = \frac{A_0}{6} - \begin{cases} 0 & 1/2 < \cos(\alpha/2) \\ \frac{A_b}{3}\frac{\cos(\alpha/2)-1/2}{\cos(\alpha/2)-1} & 1/2 \geq \cos(\alpha/2). \end{cases} \tag{5.2}$$

Figure 5.6 shows the results of this example. Both algorithms give accurate results. However, there are some exceptional values of α for which the accuracy of PANG completely drops. They appear only when $\cos(\alpha/2) \leq 1/2$, and the relative error is roughly constant for these outliers.

The computation time of PANG is clearly superior. The new algorithm includes a number of complicated elements necessary to ensure robustness. Improvements can certainly be made to streamline the calculations and comparisons. Interestingly, the computation time of the new algorithm noticeably changes after $\cos(\alpha/2) \leq 1/2$. This indicates a new regime where it must find additional intersections.

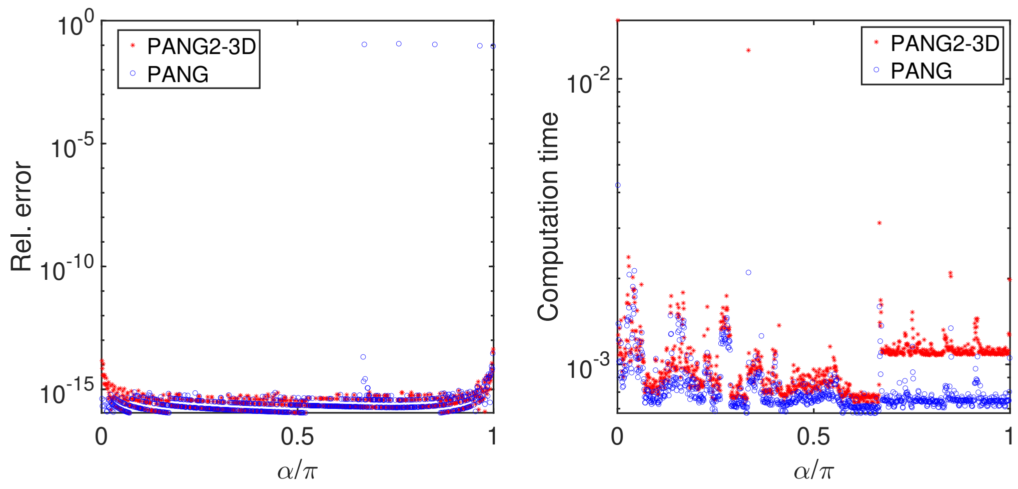


Figure 5.6: Relative error (left) and computation time (right) for this example for both PANG and PANG2-3D.

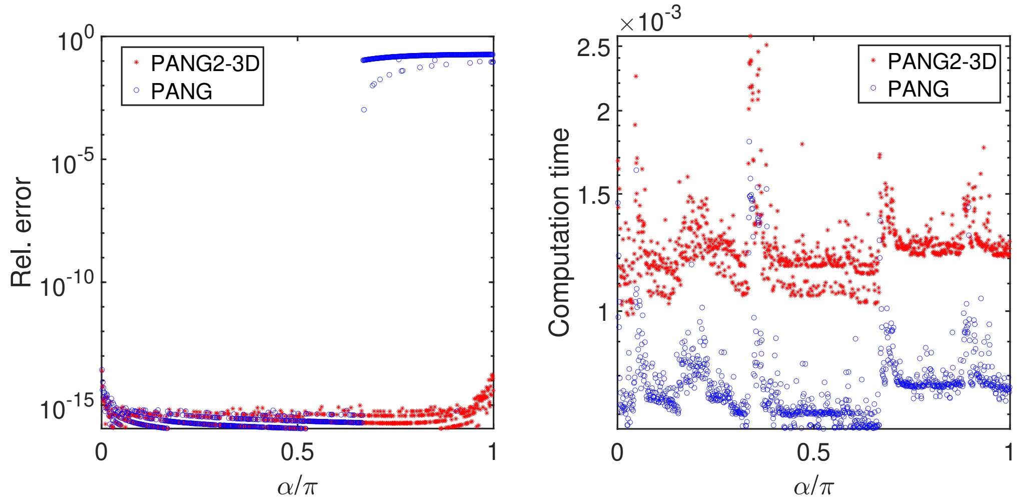


Figure 5.7: Same as Figure 5.6 but with the roles of V and U reversed.

Reversing the roles of V and U exacerbates the problem PANG has with the second regime. Now all but one of the intersections in this regime is found within a reasonable margin of the exact value. All others have error on the order of 10^{-1}. It is worth noting that there is no visible difference in the intersections calculated by these algorithms. It is only on comparison with the exact value that it becomes clear PANG is deficient.

In terms of computation time, PANG is still superior. However, the reversal of V and U may have improved the new algorithm's time, albeit minimally.

5.7 Conclusions

The intersection algorithm in the previous chapter has now been extended to 3D. It uses the same principle of parsimony, and the problem has become slightly more difficult in this higher dimension. Because the algorithm remains parsimonious, it remains robust. Errors that affect the shape of the intersection are considered to further prove this point.

The general form of this algorithm is presented in the subsequent chapter. A more rigorous proof of its robustness is provided there. It may be used as a proof for this 3D version of the algorithm.

Chapter 6

Intersection of simplices

In which one extends the previous algorithm to arbitrary dimension; investigates methodology for various aspects of the algorithm, including how to solve the intersections, how to march through neighbouring vertices and hyperplanes, how to connect the signs of the intersections to maintain consistency, and how to store the components in an efficient manner.

6.1 Notation and change of coordinates

Let n be the dimension of the space. For the purposes of this chapter we assume the vertices lie in $\mathbb{R}^n$. Each simplex then has $n+1$ vertices. Each pair of vertices defines an edge, each triple a plane, and so on.

The coordinates can no longer be easily identified as x, y and z. Instead, each coordinate is identified by the unit vector $\mathbf{e}_\gamma$ for $\gamma = 1, ..., n$, where $e_{\gamma,\eta} = \delta_{\gamma,\eta}$. In this way a given vector $\mathbf{x} \in \mathbb{R}^n$ may be written as

$$\mathbf{x} = \begin{bmatrix} \mathbf{x} \cdot \mathbf{e}_1 \\ \vdots \\ \mathbf{x} \cdot \mathbf{e}_n \end{bmatrix}.$$

Since simplices in $\mathbb{R}^n$ are bounded by $n+1$ hyperplanes, see equation (6.3, it will be useful to denote an additional coordinate, $\mathbf{x} \cdot \mathbf{e}_0$, defined as

$$\mathbf{x} \cdot \mathbf{e}_0 = 1 - \sum_{\gamma=1}^{n} \mathbf{x} \cdot \mathbf{e}_\gamma. \tag{6.1}$$

Let the two simplices to be intersected be called U and V, and let the coordinates of their vertices be denoted by $\hat{V}$ and $\hat{U}$. Let

$$\hat{V} = \mathbf{v_0}\mathbf{1}^\top + \begin{bmatrix} \mathbf{0} & \mathbf{v}_1 & \dots & \mathbf{v}_n \end{bmatrix}$$

where $\mathbf{1}$ is a column vector of $n+1$ 1's and $\mathbf{0}$ is a column vector with n 0's.

We seek to transform the coordinates into a system such that $\mathbf{v}_0$ lies at the origin and the edges of V are aligned with the unit vectors $\mathbf{e}_\gamma$ for $\gamma = 1, ..., n$. Let the transformations of U and V under this change of coordinates be represented by X and Y, respectively. The vertices of Y are then known, while the vertices of X may be found by solving the system

$$\begin{bmatrix} \mathbf{v}_1 & \dots & \mathbf{v}_n \end{bmatrix} \begin{bmatrix} \mathbf{x}_0 & \dots & \mathbf{x}_n \end{bmatrix} = \hat{U} - \mathbf{v}_0\mathbf{1}^\top. \tag{6.2}$$

The intersection of two simplices is a polytope of the same dimension. Let W be the intersection of U and V, and let Z be the intersection of X and Y and therefore the transformation of W under the same change of coordinates described above. The algorithm herein described will construct Z and then reverse the change of coordinates to arrive at W. This algorithm will be referred to as PANG2-nD.

6.2 Vertices of X inside Y

Let P_γ be an $(n-1)$–dimensional hyperplane that bounds Y, defined as

$$P_\gamma = \{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{x} \cdot \mathbf{e}_\gamma = 0\} \tag{6.3}$$

for $\gamma = 0, ..., n$. The simplex Y is then the intersection of all the half-spaces that lie on the positive sides of P_γ for all γ. In other words, a given point $\mathbf{x} \in \mathbb{R}^n$ lies inside of Y if $\mathbf{x} \cdot \mathbf{e}_\gamma \geq 0$ for all γ.

We continue to use the binary-valued sign function, see equation (4.1). The indicator function for Y may then be expressed as

$$\chi_Y(\mathbf{x}) = \prod_{\gamma=0}^{n} \operatorname{sign}(\mathbf{x} \cdot \mathbf{e}_\gamma). \tag{6.4}$$

Naturally, $\chi_Z(\mathbf{x}) = \chi_Y(\mathbf{x})\chi_X(\mathbf{x})$. Note that $\chi_X(\mathbf{x}_i) = 1$ for all $i = 0, ..., n$ and so $\mathbf{x}_i \in Z$ if and only if $\operatorname{sign}(\mathbf{x} \cdot \mathbf{e}_\gamma) = 1$ for all γ. In such a case, the i–th vertex of X lies inside both Y and Z.

6.3 Intersections between edges of X and hyperplanes of Y

The polytope Z is bounded by the intersections between X and the hyperplanes P_γ. To find the vertices of Z one must find these intersections.

The edge of X between its i–th and j–th vertices intersects P_γ if and only if $\operatorname{sign}(\mathbf{x}_i \cdot \mathbf{e}_\gamma) \neq \operatorname{sign}(\mathbf{x}_j \cdot \mathbf{e}_\gamma)$. Due to the binary nature of the sign function the number of intersections that can be calculated is strictly limited.

Proposition 6.1. *Let X be an n–simplex intersecting an $(n-1)$–hyperplane P. At least n edges of X intersect P. At most $\lceil (n+1)/2 \rceil \lfloor (n+1)/2 \rfloor$ edges of X intersect P.*

Proof Let the hyperplane P be defined by $p = 0$ for some linear function p. An n–simplex has $n+1$ vertices. Each vertex has a value of $\operatorname{sign}(p_i)$ equal to either 0 or 1. There are $\lceil (n+1)/2 \rceil$ ways to partition $n+1$ objects into two groups. Those partitionings with the largest and smallest number of pairs are listed in Table 6.1. As it is assumed that X intersects P it must be that the number of edges that intersect P is between n and $\lceil (n+1)/2 \rceil \lfloor (n+1)/2 \rfloor$. ∎

Thus, the portion of X that intersects P_γ either: does not exist; is itself a simplex in $\mathbb{R}^{n-1}$ or; is a polytope in $\mathbb{R}^{n-1}$. The intersection will then always have dimension $n-1$, unless it does not exist, which only occurs when no edges intersect P_γ. The intersection is then always consistent with the distribution of vertices of X. For example, there will never be only one intersection with P_γ, which would not be consistent with a simplex

$m(a)$	$m(b)$	pairs
$n+1$	0	0
n	1	n
$\vdots$	$\vdots$	$\vdots$
$\lceil (n+1)/2 \rceil$	$\lfloor (n+1)/2 \rfloor$	$\lceil (n+1)/2 \rceil \lfloor (n+1)/2 \rfloor$

Table 6.1: Ways to partition $n+1$ elements into two parts.

intersecting a hyperplane as this would be a 0–dimensional object in an $(n-1)$–dimensional space.

If the intersection $X \cap P_\gamma$ lies entirely within Y, which may be checked using the indicator function χ_Y, then its vertices are shared with Z. Otherwise, one must intersect this intersection with those P_η for which part of $X \cap P_\gamma$ lies in the negative half-space.

6.4 Higher dimensional intersections between X and Y

As explained above, the set of intersections between edges of X and a given hyperplane P_γ form a polytope of dimension $n-1$. That part of this polytope that lies on the positive side of the remaining hyperplanes forms part of Z. Suppose we intersect $X \cap P_\gamma$ with P_η. An edge of $X \cap P_\gamma$ intersects P_η if and only if the signs of the two attached vertices in the $\mathbf{e}_\eta$ direction differ. Proposition 6.1 applies again, ensuring a consistent number of intersections are calculated. This procedure is then repeated for this intersection, $X \cap P_\gamma \cap P_\eta$, until all hyperplanes are considered or the result lies entirely within Y.

Consider a given stage of this process where we seek $X \cap_{\gamma \in \Gamma} P_\gamma$ where Γ contains m indices within $\{0, ..., n\}$. The intersections at this stage are between this intersection of n–hyperplanes, which is itself an $(n-m)$–hyperplane, and the convex hulls of $m+1$ vertices of X, called m–faces. Each intersection is then uniquely determined by Γ, the collection of hyperplanes, and J, the set of vertices of X that form the m–face.

6.5 Calculation of intersections

Lemma 6.2. *Suppose the m–face of X between the set of $m+1$ vertices $\{\mathbf{x}_i \mid i \in J\}$ intersects the $(n-m)$–hyperplane defined as the intersection of the m hyperplanes $\{P_\gamma \mid \gamma \in \Gamma\}$. Denote this intersection as $\mathbf{q}^J_\Gamma$. Then*

$$\mathbf{q}^J_\Gamma \cdot \mathbf{e}_\eta = \frac{\begin{vmatrix} \mathbf{x}_{i_0} \cdot \mathbf{e}_\eta & \mathbf{x}_{i_0} \cdot \mathbf{e}_{\gamma_1} & \dots & \mathbf{x}_{i_0} \cdot \mathbf{e}_{\gamma_m} \\ \vdots & \vdots & & \vdots \\ \mathbf{x}_{i_m} \cdot \mathbf{e}_\eta & \mathbf{x}_{i_m} \cdot \mathbf{e}_{\gamma_1} & \dots & \mathbf{x}_{i_m} \cdot \mathbf{e}_{\gamma_m} \end{vmatrix}}{\begin{vmatrix} 1 & \mathbf{x}_{i_0} \cdot \mathbf{e}_{\gamma_1} & \dots & \mathbf{x}_{i_0} \cdot \mathbf{e}_{\gamma_m} \\ \vdots & \vdots & & \vdots \\ 1 & \mathbf{x}_{i_m} \cdot \mathbf{e}_{\gamma_1} & \dots & \mathbf{x}_{i_m} \cdot \mathbf{e}_{\gamma_m} \end{vmatrix}}. \tag{6.5}$$

Proof Without loss of generality, suppose $J = \{0, \dots, m\}$. The m–face can be defined by $\{\mathbf{g}(\{a_i\}) \mid 0 \le a_i \le 1\}$, where

$$\begin{aligned}\mathbf{g}(\{a_1,\dots,a_m\}) &= \sum_{i=1}^{m} a_i\,(\mathbf{x}_i - \mathbf{x}_0) + \mathbf{x}_0 \\ &= \sum_{i=1}^{m} a_i\mathbf{x}_i + \left(1 - \sum_{i=1}^{m} a_i\right)\mathbf{x}_0.\end{aligned}$$

The intersection $\mathbf{q}_\Gamma^J = \mathbf{g}(A)$ depends on Γ. We seek the set $A = \{a_1,\dots,a_m\}$ such that

$$\mathbf{g}(A)\cdot\mathbf{e}_\gamma = 0 \;\forall\; \gamma \in \Gamma.$$

We propose as a solution

$$\begin{aligned}a_i =& \frac{(-1)^i}{d}\begin{vmatrix}\mathbf{x}_0\cdot\mathbf{e}_{\gamma_1} & \dots & \mathbf{x}_0\cdot\mathbf{e}_{\gamma_m} \\ \vdots & & \vdots \\ \mathbf{x}_{i-1}\cdot\mathbf{e}_{\gamma_1} & \dots & \mathbf{x}_{i-1}\cdot\mathbf{e}_{\gamma_m} \\ \mathbf{x}_{i+1}\cdot\mathbf{e}_{\gamma_1} & \dots & \mathbf{x}_{i+1}\cdot\mathbf{e}_{\gamma_m} \\ \vdots & & \vdots \\ \mathbf{x}_m\cdot\mathbf{e}_{\gamma_1} & \dots & \mathbf{x}_m\cdot\mathbf{e}_{\gamma_m}\end{vmatrix}, \\ 1 - \sum_{i=1}^{m} a_i =& \frac{1}{d}\begin{vmatrix}\mathbf{x}_1\cdot\mathbf{e}_{\gamma_1} & \dots & \mathbf{x}_1\cdot\mathbf{e}_{\gamma_m} \\ \vdots & & \vdots \\ \mathbf{x}_m\cdot\mathbf{e}_{\gamma_1} & \dots & \mathbf{x}_m\cdot\mathbf{e}_{\gamma_m}\end{vmatrix}\end{aligned}$$

for some constant d. In this way the coordinates of the intersection are

$$\mathbf{g}(A)\cdot\mathbf{e}_\eta = \frac{1}{d}\begin{vmatrix}\mathbf{x}_0\cdot\mathbf{e}_\eta & \mathbf{x}_0\cdot\mathbf{e}_{\gamma_1} & \dots & \mathbf{x}_0\cdot\mathbf{e}_{\gamma_m} \\ \vdots & \vdots & & \vdots \\ \mathbf{x}_m\cdot\mathbf{e}_\eta & \mathbf{x}_m\cdot\mathbf{e}_{\gamma_1} & \dots & \mathbf{x}_m\cdot\mathbf{e}_{\gamma_m}\end{vmatrix}.$$

It is clear that if $\eta \in \Gamma$ then the coordinate is zero and $\mathbf{g}(A)$ lies on the intersection of the planes $\{P_\gamma \mid \gamma \in \Gamma\}$.

The constant d is found by rearranging the formula for $1 - \sum a_i$:

$$\begin{aligned}1 =& \frac{1}{d}\begin{vmatrix}\mathbf{x}_1\cdot\mathbf{e}_{\gamma_1} & \dots & \mathbf{x}_1\cdot\mathbf{e}_{\gamma_m} \\ \vdots & & \vdots \\ \mathbf{x}_m\cdot\mathbf{e}_{\gamma_1} & \dots & \mathbf{x}_m\cdot\mathbf{e}_{\gamma_m}\end{vmatrix} + \sum_{i=1}^{m} a_i \\ =& \sum_{i=0}^{m}\frac{(-1)^i}{d}\begin{vmatrix}\mathbf{x}_0\cdot\mathbf{e}_{\gamma_1} & \dots & \mathbf{x}_0\cdot\mathbf{e}_{\gamma_m} \\ \vdots & & \vdots \\ \mathbf{x}_{i-1}\cdot\mathbf{e}_{\gamma_1} & \dots & \mathbf{x}_{i-1}\cdot\mathbf{e}_{\gamma_m} \\ \mathbf{x}_{i+1}\cdot\mathbf{e}_{\gamma_1} & \dots & \mathbf{x}_{i+1}\cdot\mathbf{e}_{\gamma_m} \\ \vdots & & \vdots \\ \mathbf{x}_m\cdot\mathbf{e}_{\gamma_1} & \dots & \mathbf{x}_m\cdot\mathbf{e}_{\gamma_m}\end{vmatrix} \\ \implies d =& \begin{vmatrix}1 & \mathbf{x}_0\cdot\mathbf{e}_{\gamma_1} & \dots & \mathbf{x}_0\cdot\mathbf{e}_{\gamma_m} \\ \vdots & \vdots & & \vdots \\ 1 & \mathbf{x}_m\cdot\mathbf{e}_{\gamma_1} & \dots & \mathbf{x}_m\cdot\mathbf{e}_{\gamma_m}\end{vmatrix}.\end{aligned}$$

As a small validation of this formula, if $\mathbf{x}_i \cdot \mathbf{e}_\eta = c$ then $\mathbf{g}(A) \cdot \mathbf{e}_\eta = c$. ■

Corollary 6.3. *The numerator of $\mathbf{q}_\Gamma^J \cdot \mathbf{e}_\eta$ is shared with the numerators of $\mathbf{q}_{\Gamma_j}^J \cdot \mathbf{e}_j$ for m values of j, up to a change in sign, where Γ and Γ_j have cardinality m.*

Proof For each $j \in \Gamma$ define Γ_j as

$$\Gamma_j = \{\eta\} \cup \Gamma \setminus \{j\} .$$

Since Γ has m elements there are m such Γ_j. For each of these the numerator of $\mathbf{q}_{\Gamma_j}^J \cdot \mathbf{e}_j$ is the same up to an exchange of columns in the determinant. ■

By this corollary, if there is a change in sign of $\mathbf{q}_\Gamma^J \cdot \mathbf{e}_\eta$ then the entire m–face of X defined by the indices J ends up on the other side of the $(n-m)$–face of Y defined by $\Gamma \cup \{\eta\}$. This ensures consistency between intersections of a given generation. The m–face remains whole and intact. If the signs were not connected across an m–face then an error in one sign could cause distortions in the m–face, causing it to split apart.

If the J–th m–face of X does not have $m+1$ intersections then this no longer holds true. While the intersections that do exist would be consistent amongst themselves they may not agree with other calculations in the algorithm. However, if an m–face has less than $m+1$ intersections then there exist subsets of J, $J \setminus \{j\}$ and $J \setminus \{i\}$, and a subset of Γ, $\Gamma \setminus \{\gamma\}$, such that

$$\begin{aligned} \operatorname{sign}(\mathbf{q}_{\Gamma\setminus\{\gamma\}}^{J\setminus\{j\}} \cdot \mathbf{e}_\eta) &= \operatorname{sign}(\mathbf{q}_{\Gamma\setminus\{\gamma\}}^{J\setminus\{i\}} \cdot \mathbf{e}_\eta), \\ \operatorname{sign}(\mathbf{q}_{\Gamma\setminus\{\gamma\}}^{J\setminus\{j\}} \cdot \mathbf{e}_\gamma) &\neq \operatorname{sign}(\mathbf{q}_{\Gamma\setminus\{\gamma\}}^{J\setminus\{i\}} \cdot \mathbf{e}_\gamma). \end{aligned}$$

Then the sign of $\mathbf{q}_\Gamma^J \cdot \mathbf{e}_\eta$ may be determined without further calculations:

$$\operatorname{sign}(\mathbf{q}_\Gamma^J \cdot \mathbf{e}_\eta) = \operatorname{sign}(\mathbf{q}_{\Gamma\setminus\{\gamma\}}^{J\setminus\{j\}} \cdot \mathbf{e}_\eta).$$

This essentially replaces the determination of the sign of the numerator of the intersection. Such a relation exists for each intersection of the m–face. The relations between the signs within an m–face remains defined by the denominators. Thus, this sign determination still need only occur once.

6.6 Reverse change of coordinates

As noted earlier, if $\chi_Y(\mathbf{q}) = 1$ then the intersection is a vertex of Z. Therefore, its image under the reverse of the change of coordinates is a vertex of W. Since the transformation from the original coordinates is achieved by solving the system

$$\begin{bmatrix} \mathbf{v}_1 & \dots & \mathbf{v}_n \end{bmatrix} \mathbf{x} = \mathbf{u} - \mathbf{v}_0,$$

the reverse is simple multiplication by this matrix and additon by $\mathbf{v}_0$. The vertex of W is then

$$\mathbf{w}_\Gamma^J = \mathbf{v}_0 + \sum_{\gamma \notin \Gamma \cup \{0\}} \left(\mathbf{q}_\Gamma^J \cdot \mathbf{e}_\gamma\right) \cdot \mathbf{v}_\gamma. \tag{6.6}$$

Note that the 0–th coordinate of $\mathbf{q}$ is excluded from the sum as it is purely computational. Also excluded are those coordinates known to be zero, namely the set Γ.

6.7 Vertices of Y inside X

After the intersections between $(n-1)$–faces of X and edges of Y have been calculated, consider one of these edges. For this edge, there is an infinite line that extends from it such that either: there are no intersections between this line and an $(n-1)$–face of X, in which case neither vertex of Y attached to the corresponding edge lies within X, or; there are two intersections along this line. In the latter case, if the two intersections surround a vertex on the corresponding edge then that vertex lies within X.

Each of these infinite lines is the intersection of $n-1$ hyperplanes $\{P_\gamma\}_{\gamma\in\Gamma}$. Thus, for each line there remain two hyperplanes that have not intersected with this line. The intersection between this line and either of the two remaining hyperplanes is a vertex of Y.

Without loss of generality, we consider the intersection between this collection of hyperplanes Γ and the remaining hyperplane P_η. As before, there is an intersection with X if and only if the two intersections $\mathbf{q}_\Gamma^J$ and $\mathbf{q}_\Gamma^K$ have different signs in the $\mathbf{e}_\eta$ direction. Unlike in previous cases, however, any resulting intersection would lie in a 0–dimensional space, removing the need to calculate it. Thus, $\operatorname{sign}(\mathbf{q}_\Gamma^J\cdot\mathbf{e}_\eta) \neq \operatorname{sign}(\mathbf{q}_\Gamma^K\cdot\mathbf{e}_\eta)$ immediately implies the vertex at $P_\eta \cap_{\gamma\in\Gamma} P_\gamma$. Incidentally, this is the vertex at $\mathbf{e}_\nu$, such that $\nu \notin \Gamma$ and $\nu \neq \eta$.

For every vertex there are n edges that extend from it, one for each other vertex of Y. This means this test can be performed up to n times, each with a different pair of intersections. Because previous steps have ensured the configuration of these intersections around the vertex are consistent, each test will return the same result. As such, the test need only be performed once.

6.8 Algorithm

There are $n+1$ generations of intersections between m–faces of X and $(n-m)$–hyperplanes of Y, corresponding to taking between 0 and n intersections of the hyperplanes P_γ. Each combination of hyperplanes must be checked, as outlined in the above sections of this chapter. Since a given combination of m hyperplanes is used to determine the sign of an intersection in the same generation but for an unrelated combination of hyperplanes, it is necessary to complete a given generation before continuing with the next.

Algorithm 6.1 (PANG2-nD)**. Step 1: Change of coordinates.** *Find an affine transformation such that the $n+1$ vertices of V are mapped to the origin and $\mathbf{e}_j$ for $j=1,\dots,n$. Use this transformation to map U to the simplex X, see equation (6.2).*

Step 2: Vertices of X in Y. *Test if $\operatorname{sign}(\mathbf{x}_i\cdot\mathbf{e}_\gamma)=1$ for all γ, see equation (6.4). If so, $\mathbf{x}_i$ lies in Y and the i–th vertex of U in V. Repeat for all vertices of X.*

Step 3: Intersections of X with Y. *Initiate $m=0$. Let Γ denote a collection of m hyperplanes P_γ, see equation (6.3).*

Step 3 (i): Calculate the intersections. *Choose Γ and one hyperplane P_η not in this collection. For each J compare $\operatorname{sign}(\mathbf{q}_\Gamma^J\cdot\mathbf{e}_\eta)$ with that of $\operatorname{sign}(\mathbf{q}_\Gamma^K\cdot\mathbf{e}_\eta)$ for all K such that the edge between J and K lies on the boundary of X, assuming* *both $\mathbf{q}_\Gamma^J$ and $\mathbf{q}_\Gamma^K$ exist. If the signs are different, calculate $\mathbf{q}_{\Gamma\cup\{\eta\}}^{J\cup K}$ unless already computed. If the signs are the same, flag the combination $J \cup K$ and Γ, the direction η and the sign of the two.*

Repeat for all possible Γ and η combinations.

Step 3 (ii): Determine the signs. *For each combination of $J\cup K$ and $\Gamma\cup\{\eta\}$ for which an intersection was calculated in the previous step, check if the combination $J \cup K$ and $\tilde{\Gamma}$ were flagged in that same step for any $\tilde{\Gamma} \subset \Gamma \cup \{\eta\}$. If so, retrieve the noted direction ν and sign s. Then the sign in the ν direction of this intersection is equal to s.*

Otherwise, check if another intersection on the same m–face as this one has had its sign calculated. If so, determine the sign of this intersection using the knowledge that they share a numerator. Otherwise, calculate its sign as normal.

Step 3 (iii): Reverse transform *For each intersection $\mathbf{q}$ found in Step 3 (i) check $\chi_Y(\mathbf{q})$. If it equals 1 then transform $\mathbf{q}$ into original coordinates, see equation (6.6). The result is a vertex of W.*

Increment m. Repeat until $m = n - 1$.

Step 4: Vertices of Y in X. *For each vertex $\mathbf{e}_\gamma$ of Y, choose Γ a collection of $n-1$ hyperplanes that does not contain P_γ. Let the other hyperplane not in this collection be P_η. There are two intersections for this collection, J and K. If these intersections have different signs in the η direction then the vertex is in X.*

Theorem 6.4. *Algorithm 6.1 is consistent with respect to shape.*

Proof For the algorithm to be consistent with respect to shape we need to consider only two types of values of each step: the number of intersections to be calculated, and; the signs of the coordinates of these intersections. This uniquely identifies the shape of the intersection Z. The remainder of the algorithm will affect its accuracy, but not its consistency.

Consider a collection of hyperplanes Γ and a new direction $\gamma \notin \Gamma$. The number of intersections to be calculated for the collection $\Gamma \cup \{\gamma\}$ is determined directly from the number of pairs of intersections of the previous generation with collection Γ who differ in sign in the direction γ. Thus, the number of intersections is consistent with the results of the previous step of the algorithm. In Section 6.9.2 we will discuss how to restrict this test to intersections found on the boundary of the intersection Z.

Suppose $|\Gamma| = m$, then by Corollary 6.3 the sign of the numerator of an intersection is shared amongst up to $m+1$ intersections. These $m+1$ intersections represent the intersection between a given m–face of X and the collection of hyperplanes indexed by $\Gamma \cup \{\gamma\}$. Since all these intersections share this numerator, they share its sign, and their determination is self-consistent.

If this m–face of X has less than the maximum number of intersections then the signs of the numerators may be determined directly from the signs of the intersections of the $(m-1)$–faces of X that compose the m–face. Otherwise, the sign of the numerator is necessarily independent of previous calculations: If it could be determined by previous results the entire intersection Z would be found after one step of the algorithm. Therefore, the calculation of the signs of the numerators need only be consistent with the previous step in the algorithm in certain cases.

The signs of the denominators can be determined from previous steps, as will be shown in Section 6.9.3. These signs require up to two previous steps of the algorithm. Their determination is then consistent with these two previous steps. These denominators are also shared amongst all coordinates of the same intersection. Their determination is then also self-consistent within the given step.

Each step of the algorithm has been proven to be self-consistent and consistent with the previous steps. By induction, the algorithm as a whole is consistent for any dimension. ■

Note the algorithm permits paradoxical results due to the use of $\mathbf{e}_0$ as a direction for the determination of signs. The coordinates of intersections and vertices along this direction can be found using the other directions. However, doing so decouples the signs of the numerators and denominators, meaning the algorithm's steps would not be self-consistent. Since $\mathbf{e}_0$ is not used to determine the location of an intersection or vertex, we allow the paradox where an intersection can be found on the negative side of the hyperplane P_0 while its sign in the $\mathbf{e}_0$ direction is found to be positive, or vice versa. This only affects the shape of the intersection Z, and any loss in accuracy is still dictated by the formulae used.

If we apply PANG2-nD to dimension 3 and the intersection of two tetrahedra then we expect to retrieve identical results to PANG2-3D. In this particular case $\mathbf{x}_i = \begin{bmatrix} x_i & y_i & z_i \end{bmatrix}^\top$.

The first two steps of the algorithms already correspond to one another. In the third step of the general algorithm $\mathbf{q}_{\{j\}}^{\{i,k\}} \cdot \mathbf{e}_\eta$ is calculated. Suppose $j = 1$ and $\eta = 2$, then

$$\mathbf{q}_{\{1\}}^{\{i,k\}} \cdot \mathbf{e}_2 = \frac{\begin{vmatrix} y_i & x_i \\ y_k & x_k \end{vmatrix}}{\begin{vmatrix} 1 & x_i \\ 1 & x_k \end{vmatrix}}.$$

This is identical to q_x^{ik} from the tetrahedral algorithm, see equation (5.1). It is straightforward to work out that the other combinations of j and η will give the equations for q_γ^{ik}, r_γ^{ik} and $1 - q_\gamma^{ik} - r_\gamma^{ik}$, see Table 5.3. Likewise, one can show the calculations in step m, repeated once in this case, are identical to those of step 2bi of the tetrahedral algorithm. Since these calculations are the same, the comparison of the signs of the intersections will return the same results. Thus, both algorithms will agree as to which vertices of Y lie inside X.

6.9 Implementation of PANG2-nD

There are a number of practical concerns when it comes to implementing PANG2-nD. While steps 1 and 2 are straightforward, steps 3 and 4 present computational difficulties. Flowcharts of these steps are provided in Figures 6.1and 6.2.

To continue this discussion of practicalities we suppose we are on a particular iteration of the algorithm, such that we have all intersections between all m–faces of X and collections of $m-1$ hyperplanes of Y. That is, let Γ be a set of $m-1$ vertices taken from $\{0, \ldots, n\}$, and J and K similar sets of m vertices from the same choice of indices. Let γ and η be other indices from this set.

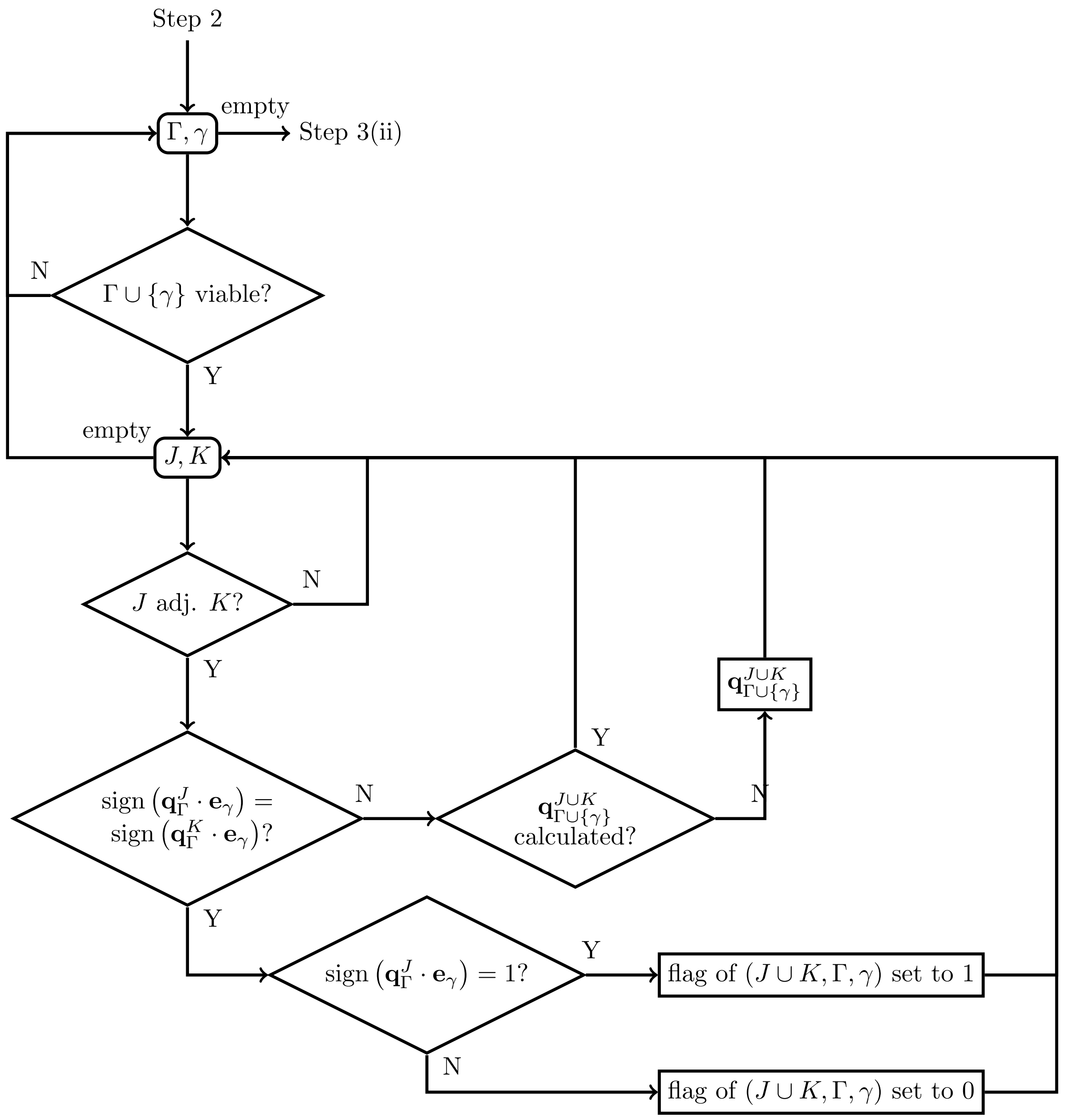

Step 2
empty
Γ, γ
Step 3(ii)
N
$\Gamma \cup \{\gamma\}$ viable?
Y
empty
J, K
N
J adj. K?
Y
$\mathbf{q}_{\Gamma\cup\{\gamma\}}^{J\cup K}$
Y
N
sign $\left(\mathbf{q}_\Gamma^J \cdot \mathbf{e}_\gamma\right) =$ sign $\left(\mathbf{q}_\Gamma^K \cdot \mathbf{e}_\gamma\right)$?
N
$\mathbf{q}_{\Gamma\cup\{\gamma\}}^{J\cup K}$ calculated?
Y
sign $\left(\mathbf{q}_\Gamma^J \cdot \mathbf{e}_\gamma\right) = 1$?
Y
flag of $(J \cup K, \Gamma, \gamma)$ set to 1
N
flag of $(J \cup K, \Gamma, \gamma)$ set to 0

Figure 6.1: Flowchart of step 3(i) of PANG2-nD.

We begin by examining Figure 6.1 and step 3(i) of PANG2-nD. The first hurdle to overcome is whether $\Gamma \cup \{\gamma\}$ is a viable option for a collection of hyperplanes. Most importantly, γ cannot lie within Γ. Other limitations can be used to narrow down the list of viable γ, as will be discussed in Section 6.9.1.

Second, one should determine whether the edge between $\mathbf{q}_\Gamma^J$ and $\mathbf{q}_\Gamma^K$ lies on the boundary of the intersection Z. If J and K are not chosen correctly, this edge may be interior and the intersection would therefore be of no interest to the algorithm. The concept of adjacency and one solution to its issues is presented in Section 6.9.2.

When the signs of the intersections in a given direction are different an intersection between them and the given hyperplane must be calculated. To save computational cost, one should then note which intersections have already been calculated, so as not to repeat unnecessary computations.

When the signs are the same, no intersection is needed. Instead this indicates the sign of any calculated intersections with the same vertex set and adjacent collections as the intersection that would have been calculated has the same sign as the two intersections at this step. Thus, those combinations are flagged in step 3(i) for simplified processing in step 3(ii).

We then proceed with step 3(ii). To maintain consistency with step 3(i) we consider the collections $\Gamma \cup \{\gamma\}$, the vertex sets $J \cup K$ and the directions η.

First, we consider only those collections Γ and directions η that have been flagged from step 3(i) for the vertex set $J \cup K$. An additional hyperplane γ must be added to the collection. It is only necessary to check for those γ such that an intersection for the combination $J \cup K$ and $\Gamma \cup \{\gamma\}$ has been calculated. One then sets $\operatorname{sign}\left(\mathbf{q}_{\Gamma\cup\{\gamma\}}^{J\cup K} \cdot \mathbf{e}_\eta\right)$ to the flagged value and computes the numerator and denominator of equation (6.5), if they haven't been already.

Once all flagged collections for $J \cup K$ have been checked, one can calculate the signs of the remaining collections $\Gamma \cup \{\gamma\}$. We choose a direction η and see if the numerator has been found from either the flagged collections or an earlier choice of $J \cup K$. If not, we take the sign of the intersection in this direction as the true sign and determine the numerator from this. The intricacies of the sign determinations are discussed in Section 6.9.3.

6.9.1 Combinations of hyperplanes

As we proceed through the algorithm we must continuously add hyperplanes to the intersections. We must somehow navigate the many combinations of these hyperplanes in an efficient manner.

Note that if there are no intersections between X and P_γ then there are no intersections between X and any collection of P_η that contains P_γ. The same is true for collections of P_η, that once there are no intersections for one collection it need not appear as a subset of any higher order collection. Any algorithm should therefore march through the set of hyperplane combinations in order of ascending dimensionality.

Each combination of hyperplanes may be arrived at through a number of pathways. For example, consider two intersecting tetrahedra in 3D. A given edge of Y lies at the intersection of two planes, P_γ and P_η. An intersection between this edge and a face of X may be found by considering either those intersections within P_γ or those within P_η. Both will indicate that an intersection is to be calculated if and only if the intersections

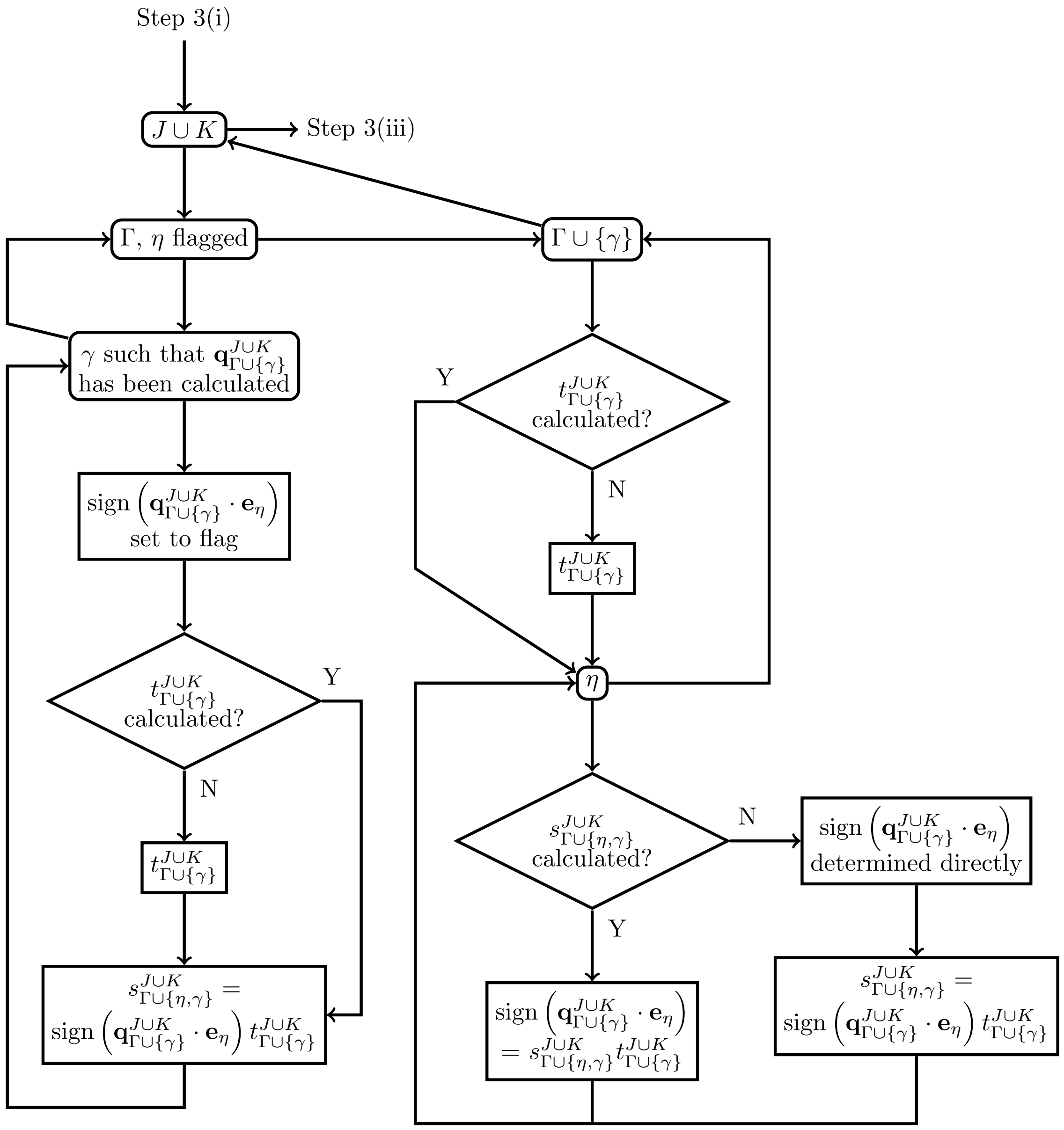

Step 3(i)
$J \cup K$
Step 3(iii)
Γ, η flagged
$\Gamma \cup \{\gamma\}$
γ such that $\mathbf{q}^{J\cup K}_{\Gamma\cup\{\gamma\}}$ has been calculated
sign $\left(\mathbf{q}^{J\cup K}_{\Gamma\cup\{\gamma\}} \cdot \mathbf{e}_\eta\right)$ set to flag
$t^{J\cup K}_{\Gamma\cup\{\gamma\}}$ calculated?
Y
N
$t^{J\cup K}_{\Gamma\cup\{\gamma\}}$
$s^{J\cup K}_{\Gamma\cup\{\eta,\gamma\}} =$ sign $\left(\mathbf{q}^{J\cup K}_{\Gamma\cup\{\gamma\}} \cdot \mathbf{e}_\eta\right) t^{J\cup K}_{\Gamma\cup\{\gamma\}}$
$t^{J\cup K}_{\Gamma\cup\{\gamma\}}$ calculated?
Y
N
$t^{J\cup K}_{\Gamma\cup\{\gamma\}}$
η
$s^{J\cup K}_{\Gamma\cup\{\eta,\gamma\}}$ calculated?
N
Y
sign $\left(\mathbf{q}^{J\cup K}_{\Gamma\cup\{\gamma\}} \cdot \mathbf{e}_\eta\right)$ determined directly
sign $\left(\mathbf{q}^{J\cup K}_{\Gamma\cup\{\gamma\}} \cdot \mathbf{e}_\eta\right) = s^{J\cup K}_{\Gamma\cup\{\eta,\gamma\}} t^{J\cup K}_{\Gamma\cup\{\gamma\}}$
$s^{J\cup K}_{\Gamma\cup\{\eta,\gamma\}} =$ sign $\left(\mathbf{q}^{J\cup K}_{\Gamma\cup\{\gamma\}} \cdot \mathbf{e}_\eta\right) t^{J\cup K}_{\Gamma\cup\{\gamma\}}$

Figure 6.2: Flowchart of step 3(ii) of PANG2-nD.

lie on opposite sides of this edge of Y, within their respective planes. As such, it is unnecessary for the purposes of identifying intersections to consider both P_γ and P_η.

However, recall that the sign of the intersection along this edge is either: connected with those of two other intersections on the same face of X or; can be determined using the signs of intersections on one of the planes attached to this edge. The latter case only occurs when this face of X does not form intersections with exactly three edges of Y. In this latter case it is necessary to check the signs of the intersections in both P_γ and P_η.

Extrapolating to higher dimensions, it is necessary to check the signs of all intersections for all collections of hyperplanes. Let Γ be a given collection of hyperplanes. When $\operatorname{sign}\left(\mathbf{q}_\Gamma^{J_1} \cdot \mathbf{e}_\gamma\right) \neq \operatorname{sign}\left(\mathbf{q}_\Gamma^{J_2} \cdot \mathbf{e}_\gamma\right)$ then one flags $\mathbf{q}_{\Gamma\cup\{\gamma\}}^{J_1\cup J_2}$ as an intersection to calculate. This flag need not be checked for any other collection $\hat{\Gamma}$ along coordinate η such that $\hat{\Gamma} \cup \{\eta\} = \Gamma \cup \{\gamma\}$, as it will return an identical result. When $\operatorname{sign}\left(\mathbf{q}_\Gamma^{J_1} \cdot \mathbf{e}_\eta\right) = \operatorname{sign}\left(\mathbf{q}_\Gamma^{J_2} \cdot \mathbf{e}_\eta\right)$ then one notes that $\operatorname{sign}\left(\mathbf{q}_{\Gamma\cup\{\gamma\}}^{J_1\cup J_2} \cdot \mathbf{e}_\eta\right) = \operatorname{sign}\left(\mathbf{q}_\Gamma^{J_1} \cdot \mathbf{e}_\eta\right)$, should such an intersection exist.

Ultimately, this means the viable set of γ for a given collection Γ in step 3(i) of PANG2-nD is all $\gamma \in \{0, \dots, n\} \setminus \Gamma$. Referring back to Figure 6.1, iterations for collections $\Gamma \cup \{\gamma\}$ that have already been checked will skip the calculation of the intersection $\mathbf{q}_{\Gamma\cup\{\gamma\}}^{J\cup K}$. The remainder of the flowchart is followed. This calculation of the intersection represents the largest computational cost of this step, but the number of hyperplane collections to check may cause the various checks to outweigh these costs.

Note that once the sign of the intersection in a given direction is established through these means even these sign checks become irrelevant. Due to the parsimony of the algorithm, if there are multiple such sign checks on the same intersection then they must agree. Perhaps more pertinent, there is no way to resolve a conflict between two such checks. Thus, the viable set of γ for a given Γ can be reduced for each such sign check.

When a sign check returns equality between two intersections, $\mathbf{q}_\Gamma^J$ and $\mathbf{q}_\Gamma^K$, in the direction η then this determines the sign of $\mathbf{q}_{\Gamma\cup\{\gamma\}}^{J\cup K}$ in the direction η for all $\gamma \notin \Gamma$. Thus, if the combination Γ and η has been flagged in one iteration, η need not be checked as a direction for any collections adjacent to Γ for the purposes of determining the signs of intersections. If $\tilde{\Gamma}$ is adjacent to Γ (see Section 6.9.2 for discussion on adjacency) then there exists $\tilde{\gamma} \notin \tilde{\Gamma}$ such that $\tilde{\Gamma} \cup \{\tilde{\gamma}\} = \Gamma \cup \{\gamma\}$ for some $\gamma \notin \Gamma$.

However, $\tilde{\Gamma} \cup \{\eta\} \neq \Gamma \cup \{\eta\}$, and so this direction still needs to be checked for the purposes of calculating intersections, unless another collection adjacent to Γ containing η has already considered the direction $\tilde{\Gamma} \setminus \Gamma$. This suggests the use of two lists of viable η for each collection Γ: one for intersection calculations, which checks if the collection $\Gamma \cup \{\eta\}$ has already produced an intersection, and; one for sign checks, which verifies if an adjacent collection $\tilde{\Gamma}$ has already found the sign of $\mathbf{q}_{\Gamma\cup\{\gamma\}}^{J\cup K}$ in the direction η. The second of these depends on the set $J \cup K$, meaning this second list interjects the flowchart between the test of adjacency and the sign check. If using such a strategy, the viability test should then be moved to this position.

6.9.2 Enumeration of combinations

Over the course of PANG2-nD one takes the combination of vertices of X and hyperplanes of Y to produce intersections. It is useful to enumerate these combinations for the purposes of storage and to keep track of certain properties that some combinations have. This enumeration is achieved with the combinatorial number system, hereafter referred to as combinadics.

Definition 6.1 (Combinadics)**.** *The ranking $N(J)$ of a subset $J = \{c_1, \ldots, c_m\}$ of natural numbers $\{0, \ldots, n\}$ is equal to*

$$N(J) = \binom{c_m}{m} + \cdots + \binom{c_1}{1}.$$

This gives to each combination of indices a position in the list of combinations. One can now associate to each set of indices J a specific row or column in a matrix indexed by $N(J)$.

This is particularly useful when considering which intersections to calculate at a given stage in the algorithm. Assuming that a set of intersections for a given collection Γ of hyperplanes forms a convex object, one need only calculate the intersections for $\Gamma \cup \{\eta\}$ for those pairs of intersections whose edges lie on the boundary of the object. These pairs have a specific relation between them, which we will use to define a notion of adjacency.

Definition 6.2 (Adjacency)**.** *Two vertex sets J and K with the same cardinality are said to be adjacent if they differ by one element. That is,*

$$|K \cap J| = |K| - 1 = |J| - 1.$$

Comparing all possible pairs of intersections will result in an excessive number of intersections, many of which will need to be discarded immediately as they lie on the inside of Z. Determining which intersections are adjacent is then necessary for an efficient implementation. If one knows only J and K then it takes $n+1$ comparisons to determine whether the two intersections are adjacent. While certainly more efficient than computing several unnecessary intersections, this may still be expensive for higher dimensions. We therefore seek a subroutine that will tell us which intersections are adjacent given which intersections were adjacent previously.

We can consider each set of indices J to be a node in a graph, with one edge to each adjacent set of indices. The graph of all calculated intersections for a given collection of hyperplanes is a subgraph of the graph of all possible intersections. This larger graph has an adjacency matrix A_m^{n+1} with each element indexed by $N(J)$ and $N(K)$, where J and K are sets of m indices chosen from $\{0, \ldots, n\}$. This element of the adjacency matrix is 1 if J and K are adjacent, and 0 if they are not. We seek then the structure of A_m^{n+1}. To simplify the notation that follows we prove results for the matrix A_m^n, which may be applied directly to the matrix A_m^{n+1}.

We first note that A_m^n is symmetric since the graph it represents is undirected. That is, J is adjacent to K if and only if K is adjacent to J. Second, it is straightforward to prove a result on the anti-transpose of A_m^n.

Definition 6.3 (Anti-transpose)**.** *The anti-transpose of a matrix A, denoted A^τ, is its reflection over its northeast-to-southwest diagonal. That is, if $A \in \mathbb{R}^{m \times m}$ then $(A^\tau)_{i,j} = (A)_{m-j-1,m-i-1}$, where $i, j = \{0, \ldots, m-1\}$.*

Lemma 6.5.

$$(A_m^n)^\tau = A_{n-m}^n. \tag{6.7}$$

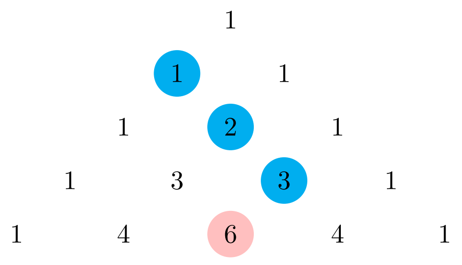


Figure 6.3: The hockey-stick identity found in Pascal's triangle. For this example, $r = 1$ and $k = 4$.

Equivalently, by the definition of anti-transpose and given that $A^n_m \in \mathbb{R}^{\binom{n}{m}\times\binom{n}{m}}$,

$$(A^n_m)_{i,j} = (A^n_{n-m})_{\binom{n}{m}-j-1,\binom{n}{m}-i-1}.$$

Proof The coordinates i and j index two sets, J_i and J_j, each with m elements out of a possible n. That is, $i = N(J_i)$ and $j = N(J_j)$. Likewise, $\binom{n}{m} - i - 1 = N(K_i)$ and $\binom{n}{m} - j - 1 = N(K_j)$ where K_i and K_j have $n - m$ elements. Then the sets K_i and K_j can be represented by those m elements not in K_i and K_j, $K_i^\complement$ and $K_j^\complement$. These sets have m elements and so their adjacency graph is a linear transformation of A^n_m. Thus, there exists a linear transform between A^n_m and A^n_{n-m}.

To prove this linear transformation is indeed the anti-transpose it suffices to prove $K_i^\complement = J_i$ and $K_j^\complement = J_j$, which may be done by showing that $N(J^\complement) = \binom{n}{m} - N(J) - 1$, or $N(J^\complement) + N(J) = \binom{n}{m} - 1$ for all J.

This is proven by induction over m. Starting with the base case $m = 1$ we have that $J = \{c_1\}$ and $J^\complement = \{0, \dots, n-1\} \setminus \{c_1\}$, leading to

$$\begin{aligned} N(J^\complement) + N(J) =& \binom{c_1}{1} + \sum_{k=0}^{c_1-1} \binom{k}{k+1} + \sum_{k=c_1+1}^{n} \binom{k}{k} \\ =& c_1 + 0 + n - (c_1 + 1) \\ =& n - 1 \\ =& \binom{n}{1} - 1. \end{aligned}$$

The statement is then true for $m = 1$.

Suppose the statement is true for m. We then verify the statement for $m+1$. Let $J = \{c_k\}_{k=1}^{m+1}$ and $J^\complement = \{\hat{c}_k\}_{k=1}^{n-m-1}$, then

$$\begin{aligned} N(J^\complement) + N(J) =& \sum_{k=1}^{m+1} \binom{c_k}{k} + \sum_{j=1}^{n-m-1} \binom{\hat{c}_j}{j} \\ =& \binom{c_{m+1}}{m+1} + \sum_{k=1}^{m} \binom{c_k}{k} + \sum_{\hat{c}_j < c_{m+1}} \binom{\hat{c}_j}{j} + \binom{c_{m+1}}{j_0} + \sum_{\hat{c}_j > c_{m+1}} \binom{\hat{c}_j}{j+1} \\ &+ \sum_{\hat{c}_j > c_{m+1}} \binom{\hat{c}_j}{j} - \sum_{\hat{c}_j > c_{m+1}} \binom{\hat{c}_j}{j+1} - \binom{c_{m+1}}{j_0} \\ =& \binom{n}{m} - 1 + \binom{c_{m+1}}{m+1} + \sum_{j=j_0}^{n-m-1} \binom{\hat{c}_j}{j} - \sum_{j=j_0}^{n-m-1} \binom{\hat{c}_j}{j+1} - \binom{c_{m+1}}{j_0}. \end{aligned}$$

Note that j_0 is the index of the first element of $J^\complement$ that is larger than c_{m+1}. In fact, since every element of $\{0, \dots, n-1\}$ is found either in J or $J^\complement$, $\hat{c}_{j_0} = c_{m+1} + 1$. Moreover, $\hat{c}_j = c_{m+1} + j - j_0 + 1$ for all $j > j_0$. In particular, $\hat{c}_{n-m-1} = c_{m+1} + n - m - j_0 = n - 1$, as long as $c_{m+1} \neq n - 1$. Therefore, $c_{m+1} = m + j_0 - 1$ and $\hat{c}_j = m + j$ for all $j > j_0$. This gives

$$\begin{aligned}
N(J^\complement) + N(J) =& \binom{n}{m} - 1 \quad + \binom{m + j_0 - 1}{m+1} + \sum_{j=j_0}^{n-m-1} \binom{m+j}{j} \\
& - \sum_{j=j_0}^{n-m-1} \binom{m+j}{j+1} - \binom{m+j_0-1}{j_0} \\
=& \binom{n}{m} - 1 \quad + \binom{m + j_0 - 1}{m+1} + \sum_{j=j_0}^{n-m-1} \binom{m+j}{j} - \sum_{j=j_0-1}^{n-m-1} \binom{m+j}{j+1} \\
=& \binom{n}{m} - 1 \quad + \binom{m + j_0 - 1}{m+1} + \sum_{j=j_0}^{n-m-1} \binom{m+j}{j} - \sum_{j=j_0}^{n-m} \binom{m+j-1}{j} \\
=& \binom{n}{m} - 1 \quad + \binom{m + j_0 - 1}{m+1} + \sum_{j=0}^{n-m-1} \binom{m+j}{j} - \sum_{j=0}^{j_0-1} \binom{m+j}{j} \\
& - \sum_{j=0}^{n-m} \binom{m+j-1}{j} + \sum_{j=0}^{j_0-1} \binom{m+j-1}{j}.
\end{aligned}$$

To simplify the sums we make use of the hockey-stick identity, see Figure 6.3:

$$\sum_{j=0}^{k-r-1} \binom{r+j}{j} = \binom{k}{r+1}.$$

In the four sums we use $r = m$ or $r = m - 1$, and $k = n$ or $k = j_0 + r$. We are then left with

$$\begin{aligned}
N(J^\complement) + N(J) =& \binom{n}{m} - 1 + \binom{m+j_0-1}{m+1} + \binom{n}{m+1} - \binom{m+j_0}{m+1} \\
& - \binom{n}{m} + \binom{m+j_0-1}{m} \\
=& \binom{n}{m+1} - 1 + \binom{m+j_0-1}{m+1} + \binom{m+j_0-1}{m} - \binom{m+j_0}{m+1} \\
=& \binom{n}{m+1} - 1,
\end{aligned}$$

where we use Pascal's rule to eliminate the terms with j_0.

	(0,3)	(1,3)	(2,3)
(0,1)	1	1	0
(0,2)	1	0	1
(1,2)	0	1	1

Table 6.2: Upper right block of the matrix A_2^4.

For the special case where $c_{m+1} = n-1$, the summations reduce to

$$\begin{aligned} N(J^\complement) + N(J) =& \binom{n}{m} + \binom{n-1}{m+1} - \binom{n-1}{m-1} - 1 \\ =& \binom{n-1}{m} + \binom{n-1}{m+1} - 1 \\ =& \binom{n}{m+1} - 1, \end{aligned}$$

making use of Pascal's rule twice. ■

As a corollary, A_m^{2m} is bisymmetric. Moreover, only half of the adjacency matrices need to be found for a given dimension n. The other half may be found by taking the anti-transpose of those already found.

For $m = 1$, there is only one index in each set. Therefore, all nodes are adjacent and the graph is a simplex. The adjacency matrix A_1^n is of size $n \times n$ with ones everywhere except along the main diagonal, which has zeros. This is true of all n.

Next, consider $m = 2$. For A_2^3 one can use Lemma 6.5 to prove $A_2^3 = A_1^3$, whose form is known. For A_2^4, we note it is bisymmetric by the same lemma. Thus, the only part missing is the upper right block, which must be persymmetric. We construct the matrix element by element. The result may be found in Table 6.2. For the remaining adjacency matrices, we must find a recurrence relation.

Lemma 6.6. *For $n > m > 1$*

$$A_m^n = \begin{bmatrix} A_m^{n-1} & \tilde{A}_m^n \\ (\tilde{A}_m^n)^\top & A_{m-1}^{n-1} \end{bmatrix}, \quad \tilde{A}_m^n = \begin{bmatrix} \tilde{A}_m^{n-1} & 0_{\binom{n-2}{m-2}\times\binom{n-2}{m}} \\ I_{\binom{n-2}{m-1}} & \tilde{A}_{m-1}^{n-1} \end{bmatrix}, \tag{6.8}$$

with starting conditions

$$A_1^n = \begin{bmatrix} 0 & 1 & \dots & 1 \\ 1 & 0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 1 \\ 1 & \dots & 1 & 0 \end{bmatrix}, \quad \tilde{A}_1^n = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}, \quad \tilde{A}_2^4 = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}. \tag{6.9}$$

Proof We divide the proof into two parts: first, we prove the diagonal blocks of A_m^n; second, we prove the structure of $\tilde{A}_m^n$. Since A_m^n is symmetric, the lower left block is the transpose of the upper right block.

Let $N(J_i)$ be the index of a column for the left blocks of A_m^n, and $N(J_j)$ the same for the right blocks. Let $N(K_i)$ be the index of a row for the upper blocks of A_m^n, and $N(K_j)$ the same for the lower blocks. Then J_i and K_i do not contain the index $n-1$, while J_j and K_j do. Then $N(J_i)$ and $N(K_i)$ are also indices for the matrix A_m^{n-1}. The relationship between J_i and K_i has not changed, and so

$$\left(A_m^{n-1}\right)_{N(J_i),N(K_i)} = \left(A_m^n\right)_{N(J_i),N(K_i)}.$$

Note that

$$\begin{aligned} N(J_j) =& \binom{n-1}{m} + \binom{c_{m-1}}{m} + \cdots + \binom{c_1}{1} \\ =& \binom{n-1}{m} + N(J_j \setminus \{n-1\}), \\ N(K_j) =& \binom{n-1}{m} + N(K_j \setminus \{n-1\}). \end{aligned}$$

Thus, the sets J_j and K_j have the same relation as $J_j \setminus \{n-1\}$ and $K_j \setminus \{n-1\}$, with their indices shifted by $\binom{n-1}{m}$. The lower right block of A_m^n is then the matrix A_{m-1}^{n-1}.

For the second part of the proof, where we consider $\tilde{A}_m^n$, we redefine the sets J_i, J_j, K_i and K_j to represent the blocks of $\tilde{A}_m^n$. Let $N(J_i)$ be the index of a column for the left blocks of $\tilde{A}_m^n$, $N(J_j)$ the same for the right blocks, $N(K_i)$ for the upper blocks and $N(K_j)$ for the lower blocks. Then

$$\begin{aligned} n-1 \in& J_i, & n-2 \notin& J_i, \\ n-1, n-2 \in& J_j, & & \\ & & n-1, n-2 \notin& K_i, \\ n-2 \in& K_j, & n-1 \notin& K_j. \end{aligned}$$

Since K_i contains neither $n-2$ nor $n-1$ and J_j contains both, there are no adjacent sets in this block. This block is then a zero matrix of size $\binom{n-2}{m} \times \binom{n-2}{m-2}$. The number of rows corresponds to the number of possible K_i, and the number of columns to the number of possible J_j.

The sets J_i and K_j differ by at least one element, since the former has $n-1$ and not $n-2$ and the latter $n-2$ and not $n-1$. Thus, the two sets are adjacent if and only if $J_i \setminus \{n-1\} = K_j \setminus \{n-2\}$, and therefore $N(J_i \setminus \{n-1\}) = N(K_j \setminus \{n-2\})$. The lower left block of $\tilde{A}_m^n$ is then the identity matrix with $\binom{n-2}{m-1}$ rows and columns, the number of possible values of $N(J_i \setminus \{n-1\})$.

For the upper left block, consider a new set $J_k = J_i \setminus \{n-1\} \cup \{n-2\}$. Then J_i is adjacent to K_i if and only if J_k is adjacent to K_i, since K_i contains neither $n-2$ nor $n-1$. Thus,

$$\left(A_m^n\right)_{N(J_i),N(K_i)} = \left(A_m^{n-1}\right)_{N(J_k),N(K_i)}.$$

The pair $N(J_k)$, $N(K_i)$ index an element in the block $\tilde{A}_m^{n-1}$, and so the upper left block of $\tilde{A}_m^n$ is a copy of $\tilde{A}_m^{n-1}$.

Since both J_j and K_j contain $n-2$, these two sets are adjacent if and only if $J_j \setminus \{n-1\}$ and $K_j \setminus \{n-2\}$ are adjacent. These sets index an element in A_{m-1}^{n-1}, specifically in the block $\tilde{A}_{m-1}^{n-1}$. Note that $N(J_j) = \binom{n-1}{m} + N(J_j \setminus \{n-1\})$ and $N(K_j) = \binom{n-2}{m} + N(K_j \setminus \{n-2\})$, and so the lower right block of $\tilde{A}_m^n$ is the block $\tilde{A}_{m-1}^{n-1}$. ■

$N(J)$	(0,1)	(0,2)	(1,2)	(0,3)	(1,3)	(2,3)
0		(0,1,2)	(0,1,2)	(0,1,3)	(0,1,3)	
1	0		(0,1,2)	(0,2,3)		(0,2,3)
2	0	0			(1,2,3)	(1,2,3)
3	1	2			(0,1,3)	(0,2,3)
4	1		3	1		(1,2,3)
5		2	3	2	3	

Table 6.3: Intergenerational table for the union of two sets. The table is naturally symmetric, with the lower triangle displaying the ranking and the upper triangle displaying the corresponding combination. The table may also be read as the union of one set and one index, where the added index is that part of the second set not found within the first.

Note that for the special case of A_m^{2m}, which is bisymmetric by Lemma 6.5, the matrix has the form

$$A_m^{2m} = \begin{bmatrix} A_m^{2m-1} & \tilde{A}_m^{2m} \\ (\tilde{A}_m^{2m})^\top & (A_m^{2m-1})^\tau \end{bmatrix}, \quad \tilde{A}_m^{2m} = \begin{bmatrix} \tilde{A}_m^{2m-1} & 0 \\ I & (\tilde{A}_m^{2m-1})^\tau \end{bmatrix}.$$

Recall we require A_m^{n+1} for the intersection of simplices in n dimensions. We may construct this matrix with the following subroutine.

Algorithm 6.2. *1. Construct A_1^{n+1}, which contains all A_1^k for $k < n+1$ as the upper left most block of size $k \times k$.*

2. Construct A_2^4 using the known matrices A_2^3 and $\tilde{A}_2^4$. Construct $\tilde{A}_2^5$ and use it to build A_2^5. Use the recurrence relations of Lemma 6.6 to construct A_2^{n+1}. Seed the construction of A_3^{n+1} with $A_3^5 = (A_2^5)^\tau$ and $\tilde{A}_3^5 = (\tilde{A}_2^5)^\tau$. Set $k = 3$.

3. Construct $\tilde{A}_k^{2k}$ and A_k^{2k}. Proceed with the recurrence to build A_k^{n+1}. Seed the construction of A_{k+1}^{n+1} with $A_{k+1}^{2k+1} = (A_k^{2k+1})^\tau$ and $\tilde{A}_{k+1}^{2k+1} = (\tilde{A}_k^{2k+1})^\tau$. Increment k and repeat this step until $k > (n+1)/2$.

Ideally, combinadics would remove the need to store the vertex sets J and the collections Γ. However, to do this we would require knowledge of certain relations of the rankings between generations. Specifically, we would require

$$N(J \cup K) = f(N(J), N(K)),$$
$$N(\Gamma \cup \{\gamma\}) = g(N(\Gamma), \gamma)$$

for J adjacent to K and $\gamma \notin \Gamma$.

These relations could be produced during runtime of PANG2-nD by constructing an intergenerational table between the rankings. A sample table is given in Table 6.3 for the $n = 3$ case with $|J| = 2$. Since the calculated entries of this table lie exactly at the positions of the entries of the corresponding adjacency matrix, the construction of the intergenerational table would replace the need for the adjacency matrix. This is most likely costly and ill-advised, as the entries of this table are only required when an intersection is found. Calculating the entire table is therefore unnecessary.

6.9.3 Connectivity of signs

As has been explained in detail in Section 6.5 those intersections of a given m–face of X and a given collection of hyperplanes of Y are connected in the signs of their components. This connectivity is what lends the algorithm its consistency of shape. This connectivity was presented above through what is effectively Cramer's rule used to solve the intersections. The numerators and denominators then have specific connections with one another. However, this represents a particularly inaccurate, unstable and inefficient manner in calculating the intersections, as will be discussed in more detail in Section 6.9.4. It is then desirable to maintain the connectivity of the Cramer's rule representation while using better subroutines to calculate the intersections.

Denote the sign of the numerator of $\mathbf{q}_\Gamma^J \cdot \mathbf{e}_\eta$ as $s_{\Gamma\cup\{\eta\}}^J$, and the sign of the denominator as t_Γ^J. Note that while in general we use the binary-valued sign function defined in equation (4.1), the values of $s_{\Gamma\cup\{\eta\}}^J$ and t_Γ^J are ± 1 so that they may be multiplied together. Alternatively, one can keep the binary-valued sign convention, but for these numbers $0^2 = 1$.

It is clear that t_Γ^J is unique to $\mathbf{q}_\Gamma^J$ while $s_{\Gamma\cup\{\eta\}}^J$ is shared, up to a single change in sign, with m other choices of $\Gamma \cup \{\eta\}$, see Corollary 6.3. If t_Γ^J experiences a swap then all signs of $\mathbf{q}_\Gamma^J$ are likewise swapped. If $s_{\Gamma\cup\{\eta\}}^J$ experiences a swap then all signs of $\mathbf{q}_{\tilde{\Gamma}}^J \cdot \mathbf{e}_\gamma$ are swapped for all $\tilde{\Gamma} \cup \{\gamma\} = \Gamma \cup \{\eta\}$.

There exist connections between the denominators of one generation of intersections and the numerators of the previous generations. These connections will be important to know for the total connectivity of the signs. For the sake of notation let

$$X_J = \begin{bmatrix} \mathbf{x}_{i_0} & \dots & \mathbf{x}_{i_m} \end{bmatrix}, \qquad\qquad I_\Gamma = \begin{bmatrix} \mathbf{e}_{\gamma_1} & \dots & \mathbf{e}_{\gamma_m} \end{bmatrix}$$

for $J = \{i_j\}_{j=0}^m$ and $\Gamma = \{\gamma_j\}_{j=1}^m$. Then

$$\mathbf{q}_\Gamma^J \cdot \mathbf{e}_\eta = \frac{\left| X_J^\top \mathbf{e}_\eta \quad X_J^\top I_\Gamma \right|}{\left| \mathbf{1} \quad X_J^\top I_\Gamma \right|}.$$

Lemma 6.7. *Suppose* $\operatorname{sign}\left(\mathbf{q}_\Gamma^{J\setminus\{i\}} \cdot \mathbf{e}_\eta\right) \neq \operatorname{sign}\left(\mathbf{q}_\Gamma^{J\setminus\{j\}} \cdot \mathbf{e}_\eta\right)$ *and an intersection* $\mathbf{q}_{\Gamma\setminus\{\gamma\}}^{J\setminus\{i,j\}}$ *was calculated for some* $\gamma \in \Gamma$, *then*

$$\begin{aligned} \operatorname{sign}\left(\left|\mathbf{1} \quad X_J^\top I_{\Gamma\cup\{\eta\}}\right|\right) &= \operatorname{sign}\left(\left|X_{J\setminus\{i\}}^\top I_{\Gamma\cup\{\eta\}}\right| \left|X_{J\setminus\{i,j\}}^\top I_\Gamma\right| \left|\mathbf{1} \quad X_{J\setminus\{j\}}^\top I_\Gamma\right|\right), \\ t_{\Gamma\cup\{\eta\}}^J &= (-1)^\eta s_{\Gamma\cup\{\eta\}}^{J\setminus\{i\}} s_\Gamma^{J\setminus\{i,j\}} t_\Gamma^{J\setminus\{j\}}. \end{aligned}$$

Proof Without loss of generality, suppose $\mathbf{x}_i$ is the first column of $X_{J\setminus\{j\}}$ and likewise $\mathbf{x}_j$ is the first column of $X_{J\setminus\{i\}}$. Furthermore, to adhere to the notation already established the first column of $I_{\Gamma\cup\{\eta\}}$ must be $\mathbf{e}_\eta$.

By assumption $\mathbf{q}_\Gamma^{J\setminus\{i\}} \cdot \mathbf{e}_\eta - \mathbf{q}_\Gamma^{J\setminus\{j\}} \cdot \mathbf{e}_\eta$ has the same sign as $\mathbf{q}_\Gamma^{J\setminus\{i\}} \cdot \mathbf{e}_\eta$. We begin by simplifying this expression:

$$\begin{aligned} \mathbf{q}_\Gamma^{J\setminus\{i\}} \cdot \mathbf{e}_\eta - \mathbf{q}_\Gamma^{J\setminus\{j\}} \cdot \mathbf{e}_\eta &= \frac{\left|X_{J\setminus\{i\}}^\top I_{\Gamma\cup\{\eta\}}\right|}{\left|\mathbf{1} \quad X_{J\setminus\{i\}}^\top I_\Gamma\right|} - \frac{\left|X_{J\setminus\{j\}}^\top I_{\Gamma\cup\{\eta\}}\right|}{\left|\mathbf{1} \quad X_{J\setminus\{j\}}^\top I_\Gamma\right|} \\ &= \frac{\left|\mathbf{1} \quad X_{J\setminus\{j\}}^\top I_\Gamma\right| \left|X_{J\setminus\{i\}}^\top I_{\Gamma\cup\{\eta\}}\right| - \left|\mathbf{1} \quad X_{J\setminus\{i\}}^\top I_\Gamma\right| \left|X_{J\setminus\{j\}}^\top I_{\Gamma\cup\{\eta\}}\right|}{\left|\mathbf{1} \quad X_{J\setminus\{i\}}^\top I_\Gamma\right| \left|\mathbf{1} \quad X_{J\setminus\{j\}}^\top I_\Gamma\right|}. \end{aligned}$$

We expand the numerator of this expression:

$$\begin{aligned}
&\left(\left|X_{J\setminus\{i,j\}}^\top I_\Gamma\right| + \sum_{k=0}^{m-1}(-1)^k \mathbf{x}_i^\top \mathbf{e}_{\gamma_k}\left|\mathbf{1} \quad X_{J\setminus\{i,j\}}^\top I_{\Gamma\setminus\{\gamma_k\}}\right|\right)\left|X_{J\setminus\{i\}}^\top I_{\Gamma\cup\{\eta\}}\right| \\
&- \left(\left|X_{J\setminus\{i,j\}}^\top I_\Gamma\right| + \sum_{k=0}^{m-1}(-1)^k \mathbf{x}_j^\top \mathbf{e}_{\gamma_k}\left|\mathbf{1} \quad X_{J\setminus\{i,j\}}^\top I_{\Gamma\setminus\{\gamma_k\}}\right|\right)\left|X_{J\setminus\{j\}}^\top I_{\Gamma\cup\{\eta\}}\right| \\
&= \left|X_{J\setminus\{i,j\}}^\top I_\Gamma\right| \begin{vmatrix} 1 & \mathbf{x}_i^\top I_{\Gamma\cup\{\eta\}} \\ 1 & \mathbf{x}_j^\top I_{\Gamma\cup\{\eta\}} \\ \mathbf{0} & X_{J\setminus\{i,j\}}^\top I_{\Gamma\cup\{\eta\}} \end{vmatrix} + \begin{vmatrix} \mathbf{x}_i^\top I_\Gamma \mathbf{w} & \mathbf{x}_i^\top I_{\Gamma\cup\{\eta\}} \\ \mathbf{x}_j^\top I_\Gamma \mathbf{w} & \mathbf{x}_j^\top I_{\Gamma\cup\{\eta\}} \\ \mathbf{0} & X_{J\setminus\{i,j\}}^\top I_{\Gamma\cup\{\eta\}} \end{vmatrix} \\
&= \left|X_{J\setminus\{i,j\}}^\top I_\Gamma\right| \begin{vmatrix} 1 & \mathbf{x}_i^\top I_{\Gamma\cup\{\eta\}} \\ 1 & \mathbf{x}_j^\top I_{\Gamma\cup\{\eta\}} \\ \mathbf{0} & X_{J\setminus\{i,j\}}^\top I_{\Gamma\cup\{\eta\}} \end{vmatrix} + \begin{vmatrix} 0 & \mathbf{x}_i^\top I_{\Gamma\cup\{\eta\}} \\ 0 & \mathbf{x}_j^\top I_{\Gamma\cup\{\eta\}} \\ -X_{J\setminus\{i,j\}}^\top I_\Gamma \mathbf{w} & X_{J\setminus\{i,j\}}^\top I_{\Gamma\cup\{\eta\}} \end{vmatrix} \\
&= \left|X_{J\setminus\{i,j\}}^\top I_\Gamma\right| \left|\mathbf{1} \quad X_J^\top I_{\Gamma\cup\{\eta\}}\right|,
\end{aligned}$$

where $w_k = (-1)^k \left|\mathbf{1} \quad X_{J\setminus\{i,j\}}^\top I_{\Gamma\setminus\{\gamma_k\}}\right|$. To prove the last equality note that each element of $X_{J\setminus\{i,j\}}^\top I_\Gamma \mathbf{w}$ is equal to the same value:

$$\begin{aligned}
\left(X_{J\setminus\{i,j\}}^\top I_\Gamma \mathbf{w}\right)_l &= \sum_{k=0}^{m-1}(-1)^k \mathbf{x}_l^\top \mathbf{e}_{\gamma_k}\left|\mathbf{1} \quad X_{J\setminus\{i,j\}}^\top I_{\Gamma\setminus\{\gamma_k\}}\right| \\
&= \begin{vmatrix} 0 & \mathbf{x}_l^\top I_\Gamma \\ \mathbf{1} & X_{J\setminus\{i,j\}}^\top I_\Gamma \end{vmatrix} \\
&= \begin{vmatrix} -1 & \mathbf{0}^\top \\ \mathbf{1} & X_{J\setminus\{i,j\}}^\top I_\Gamma \end{vmatrix} \\
&= -\left|X_{J\setminus\{i,j\}}^\top I_\Gamma\right|.
\end{aligned}$$

The simplified expression is then

$$\mathbf{q}_\Gamma^{J\setminus\{i\}} \cdot \mathbf{e}_\eta - \mathbf{q}_\Gamma^{J\setminus\{j\}} \cdot \mathbf{e}_\eta = \frac{\left|X_{J\setminus\{i,j\}}^\top I_\Gamma\right| \left|\mathbf{1} \quad X_J^\top I_{\Gamma\cup\{\eta\}}\right|}{\left|\mathbf{1} \quad X_{J\setminus\{i\}}^\top I_\Gamma\right| \left|\mathbf{1} \quad X_{J\setminus\{j\}}^\top I_\Gamma\right|}.$$

Therefore, the sign of $\left|\mathbf{1} \quad X_J^\top I_{\Gamma\cup\{\eta\}}\right|$ is

$$\operatorname{sign}\left(\left|\mathbf{1} \quad X_J^\top I_{\Gamma\cup\{\eta\}}\right|\right) = \operatorname{sign}\left(\left|X_{J\setminus\{i\}}^\top I_{\Gamma\cup\{\eta\}}\right| \left|X_{J\setminus\{i,j\}}^\top I_\Gamma\right| \left|\mathbf{1} \quad X_{J\setminus\{j\}}^\top I_\Gamma\right|\right).$$

■

Unfortunately, there is no guarantee that $\mathbf{q}_{\Gamma\setminus\{\gamma\}}^{J\setminus\{i,j\}}$ was calculated for a given step. It is tempting to suggest that the sign checks in step 3(i) provide us with information on the 'non-intersection' $\mathbf{q}_{\Gamma\setminus\{\gamma\}}^{J\setminus\{i,j\}}$, which in turn may give us information on the sign of the denominator. However, this sign check assumes that this intersection lies between two intersections of the previous generation. If this is not the case, such as with this 'non-intersection', then the sign check does not provide the necessary information.

Lemma 6.8. *Suppose* $\operatorname{sign}\left(\mathbf{q}_\Gamma^{J\setminus\{i\}} \cdot \mathbf{e}_\gamma\right) = \operatorname{sign}\left(\mathbf{q}_\Gamma^{J\setminus\{j\}} \cdot \mathbf{e}_\gamma\right)$ *for all* $\gamma \notin \Gamma$. *Then, for all* $\eta \notin \Gamma$,

$$s_{\Gamma\cup\{\gamma,\eta\}}^J = t_\Gamma^{J\setminus\{i\}} t_\Gamma^{J\setminus\{j\}} s_\Gamma^{J\setminus\{i,j\}} r_{\Gamma,\gamma,\eta}^{J,i,j}$$

where

$$r^{J,i,j}_{\Gamma,\gamma,\eta} = \begin{cases} 1 & \left(\mathbf{q}_\Gamma^{J\setminus\{i\}}\cdot\mathbf{e}_\eta\right)\left(\mathbf{q}_\Gamma^{J\setminus\{j\}}\cdot\mathbf{e}_\gamma\right) \geq \left(\mathbf{q}_\Gamma^{J\setminus\{i\}}\cdot\mathbf{e}_\gamma\right)\left(\mathbf{q}_\Gamma^{J\setminus\{j\}}\cdot\mathbf{e}_\eta\right), \\ -1 & \textit{otherwise.} \end{cases}$$

Proof Consider the plane $(\mathbf{e}_\eta \times \mathbf{e}_\gamma)\cdot\mathbf{x} = 0$ and the two intersections $\mathbf{q}_\Gamma^{J\setminus\{i\}}$ and $\mathbf{q}_\Gamma^{J\setminus\{j\}}$. The intersection between the line connecting these two intersections and the projection of the hyperplane P_γ onto this plane satisfies

$$\begin{aligned}\mathbf{q}\cdot\mathbf{e}_\eta &= \frac{\begin{vmatrix}\mathbf{q}_\Gamma^{J\setminus\{i\}}\cdot\mathbf{e}_\eta & \mathbf{q}_\Gamma^{J\setminus\{i\}}\cdot\mathbf{e}_\gamma \\ \mathbf{q}_\Gamma^{J\setminus\{j\}}\cdot\mathbf{e}_\eta & \mathbf{q}_\Gamma^{J\setminus\{j\}}\cdot\mathbf{e}_\gamma\end{vmatrix}}{\begin{vmatrix}1 & \mathbf{q}_\Gamma^{J\setminus\{i\}}\cdot\mathbf{e}_\gamma \\ 1 & \mathbf{q}_\Gamma^{J\setminus\{j\}}\cdot\mathbf{e}_\gamma\end{vmatrix}} \\ &= \begin{vmatrix}\mathbf{q}_\Gamma^{J\setminus\{i\}}\cdot\mathbf{e}_\eta & \mathbf{q}_\Gamma^{J\setminus\{i\}}\cdot\mathbf{e}_\gamma \\ \mathbf{q}_\Gamma^{J\setminus\{j\}}\cdot\mathbf{e}_\eta & \mathbf{q}_\Gamma^{J\setminus\{j\}}\cdot\mathbf{e}_\gamma\end{vmatrix} \frac{\left|\mathbf{1} \quad X_{J\setminus\{i\}}^\top I_\Gamma\right|\left|\mathbf{1} \quad X_{J\setminus\{j\}}^\top I_\Gamma\right|}{\left|X_{J\setminus\{i,j\}}^\top I_\Gamma\right|\left|\mathbf{1} \quad X_J^\top I_{\Gamma\cup\{\eta\}}\right|}\end{aligned}$$

by Lemma 6.7. Since $J\setminus\{i\}$ and $J\setminus\{j\}$ are adjacent, this is equal to $\mathbf{q}^J_{\Gamma\cup\{\gamma\}}\cdot\mathbf{e}_\eta$. Therefore,

$$\left|X_J^\top I_{\Gamma\cup\{\eta,\gamma\}}\right| = \begin{vmatrix}\mathbf{q}_\Gamma^{J\setminus\{i\}}\cdot\mathbf{e}_\eta & \mathbf{q}_\Gamma^{J\setminus\{i\}}\cdot\mathbf{e}_\gamma \\ \mathbf{q}_\Gamma^{J\setminus\{j\}}\cdot\mathbf{e}_\eta & \mathbf{q}_\Gamma^{J\setminus\{j\}}\cdot\mathbf{e}_\gamma\end{vmatrix} \frac{\left|\mathbf{1} \quad X_{J\setminus\{i\}}^\top I_\Gamma\right|\left|\mathbf{1} \quad X_{J\setminus\{j\}}^\top I_\Gamma\right|}{\left|X_{J\setminus\{i,j\}}^\top I_\Gamma\right|}$$

which leads immediately to the statement of the lemma. ■

The values of $s^J_{\Gamma\cup\{\gamma\}}$ can now be built up for all J, regardless of whether or not an appropriate intersection has been calculated. This allows Lemma 6.7 to be used for all J, even those for which $J\setminus\{i,j\}$ did not have an intersection.

Using these two lemmas in the algorithm adds further parsimony. The denominator's sign can now be found without additional calculations.

6.9.4 Alternative calculation of the intersections

The actual process of calculating $\mathbf{q}^J_\Gamma$ should not be done using the formulas presented here, as these are effectively Cramer's rule and therefore prone to error. The form presented here is merely to illustrate the connection between the signs of the intersections at various stages.

As this is the most numerically intensive part of PANG2-nD it is crucial to use a fast and, ideally, accurate method to perform this calculation. The accuracy of this step will not affect the robustness of the overall algorithm since this relies only on the consistency between signs. This consistency may (and should) be determined externally to the magnitude of the intersection.

One should consider adapting GMRES to find this solution. As an aside, one can represent $\mathbf{q}^J_\Gamma$ as

$$\mathbf{q}^J_\Gamma = X_J\mathbf{u}$$

where X_J are the positions of the J vertices of X and

$$\begin{bmatrix}\mathbf{1}^\top \\ I_\Gamma^\top X_J\end{bmatrix}\mathbf{u} = \begin{bmatrix}1 \\ \mathbf{0}\end{bmatrix}$$

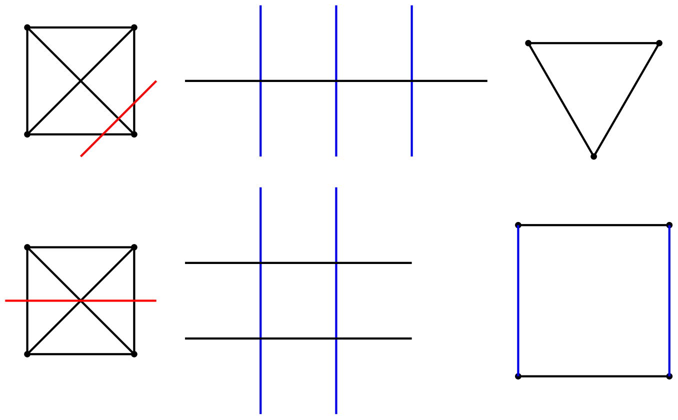


Figure 6.4: Simplicial sections of a tetrahedron.

where I_Γ has columns $\mathbf{e}_\gamma$ for $\gamma \in \Gamma$.

We could also calculate numerators and denominators separately and combine them as needed.

6.10 Visualization

Throughout the process of developing PANG2-nD it has been useful to have specific examples of various intersections. These have proven challenging to visualize due to the high dimensionality at play. I present one technique I used in this aspect, which I have taken to calling cross-hatching.

As explained in detail throughout this chapter, an intersection is found if and only if its parents fall on opposite sides of a hyperplane. At the first step of PANG2-nD these parents are the vertices of the simplex X. They may then be divided into two groups, those that lie on the positive side of the hyperplane and those on the negative.

For each positive vertex, make a vertical line. For each negative vertex, make a horizontal line that intersects all vertical lines. These lines should be evenly spaced.

Each intersection between these lines, called a cross-hatch, represents an intersection calculated at this first step of the algorithm. Each of these intersections is adjacent to all others along the same vertical and horizontal lines. Thus, each of these lines represents a simplex of a given dimension.

Figures 6.4, 6.5 and 6.6 show the results of using a hyperplane to cut through a simplex. The result of such a cut is called a simplicial section.

Note that for all dimensions two of the simplicial sections are made evident by the cross-hatching diagrams. If the hyperplane separates a single vertex from the others, the section is necessarily a simplex of the next highest dimension since all intersections fit on a single horizontal line of the cross-hatching.

If the hyperplane separates two vertices, the section is a simplicial prism, with each layer of the prism being a simplex of the second next highest dimension. This is because there are two horizontal lines, each representing a simplex, with vertical lines representing the edges that form the prism.

For example, in 4D there are only two possible sections: a tetrahedron, the simplex in 3D, and; a triangular prism, with each cut through the prism giving the simplex in 2D.

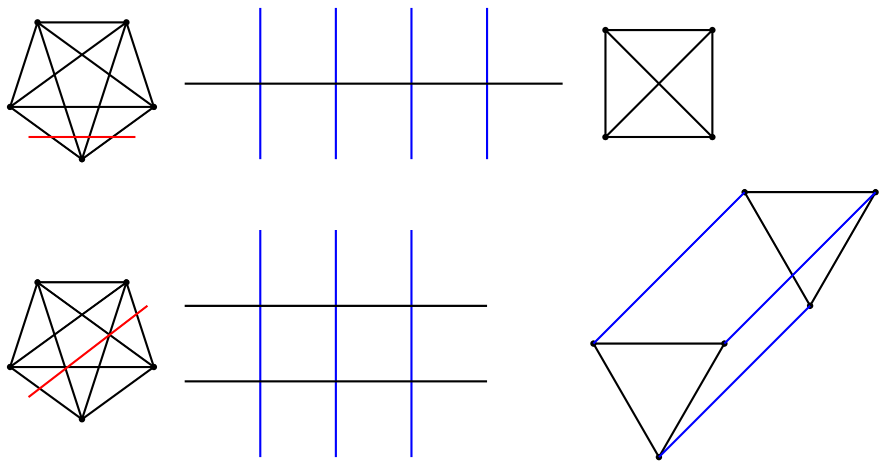


Figure 6.5: Simplicial sections of a 5-cell.

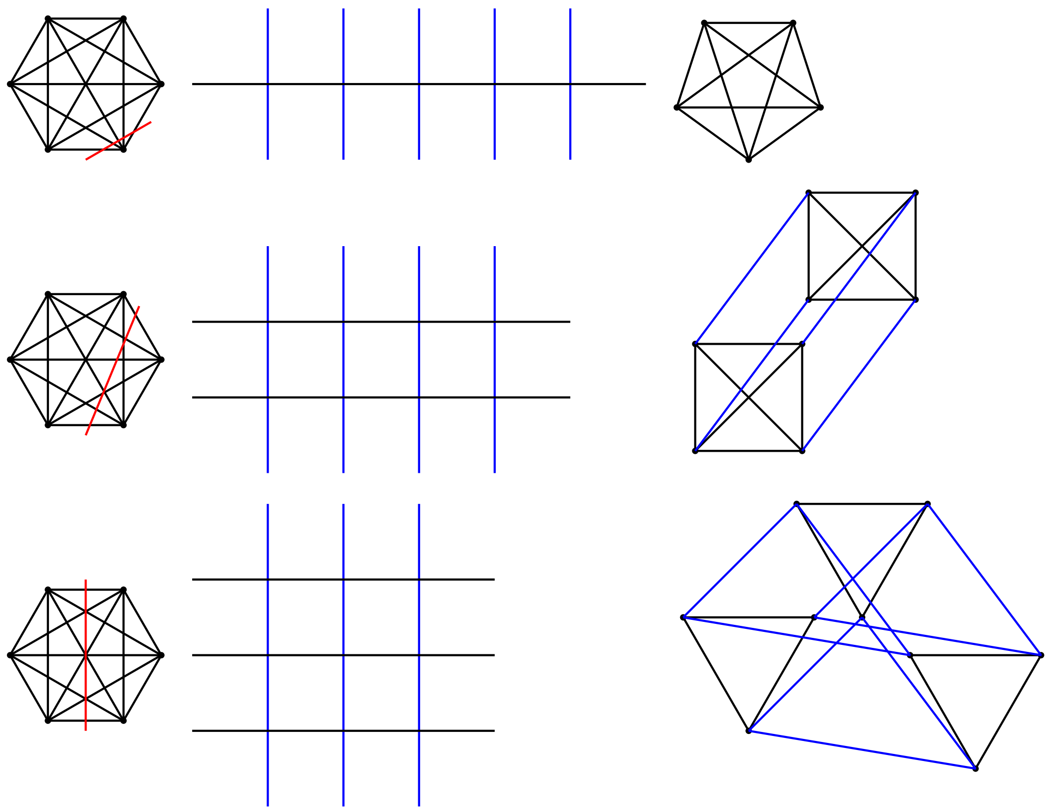


Figure 6.6: Simplicial sections of a 5-simplex. In order, they are: the 4-simplex, also called the 5-cell; the tetrahedral prism and; the 3-3 duoprism, also called the triangular duoprism.

The cross-hatching diagrams are orthogonal projections of the sections into a 2D space. Each cross-hatch is a vertex of the section. Each line represents all edges connecting all vertices that lie upon it. Since there are multiple edges for each line, it is not immediately evident what shape results from each diagram.

However, a cross-hatching diagram does indicate the exact polytope of the section. If there are n vertical lines and m horizontal lines then the section is the Cartesian product between an n–simplex and an m–simplex. It has already been explained that each vertical line represents an n–simplex. Thus, as one moves left or right along the horizontal lines one remains in an n–simplex. Thus, the cross-hatching diagrams represent duoprisms, implying that all cross-sections of simplices are duoprisms.

The cross-hatching diagrams can then help us visualize the second step of the intersection calculations. Take a second hyperplane and use it to divide the intersections of the first generation. Draw a circle on each cross-hatch that represents an intersection on the positive side of the hyperplane. Since X and Y are convex, these circles can be organized such that they form a simply connected group. For each circle, there is a new intersection of the second generation for each cross-hatch along the same horizontal and vertical lines. Two intersections are adjacent in this generation if they share a parent, or if their parents form a rectangle, see Lemma 6.10. Without loss of generality, let the circles be the fathers of the intersections.

Figure 6.7 gives all possible sections of the triangular prism, itself a section of the simplex in 4D. The other possible section of the simplex in 4D is the simplex in 3D, which has already been considered in Figure 6.4. Since this is the second step of intersecting simplices in 4D, the section is naturally a polygon.

When moving to higher dimensions, it becomes more challenging to draw the resulting section from the cross-hatching. To help, I present a few tips. Note that each circle (father) represents a simplex in the section. For example, if there are three intersections found for a given circle then the section will have a triangular face. Likewise, each empty cross-hatch that results in an intersection is a mother, and therefore represents another simplex. Start with the fathers' simplices, then add the mothers', and finally make the remaining connections between cousins.

Figure 6.8 gives an example of one second order section of the 5-simplex, specifically a section of the 3-3 duoprism. Due to the high dimensionality of the 3-3 duoprism, it is difficult to visualize a hyperplane intersecting it. Employing the cross-hatching diagrams one can retrieve the necessary information to construct the given section.

Lemma 6.9. *The vertices of a $(n-k)$–th order section of an n–simplex have exactly k edges extending from them.*

Proof Each edge of an n–polytope is shared between $n-1$ 2–faces. A dividing hyperplane that cuts an edge necessarily cuts all of the faces attached to it. Since each face is a convex polygon, this cut results in exactly two vertices of the subsequent section which are neighbours. Thus, a vertex in a section has one edge for each face that contained the edge the vertex came from. Each vertex in the section of an n–polytope then has $n-1$ edges that extend from it.

An $(n-k-1)$–th order section of an n–simplex is a $(k+1)$–polytope. Taking a section of it results in an $(n-k)$–th order section, a k–polytope, such that each vertex has k edges extending from it. ■

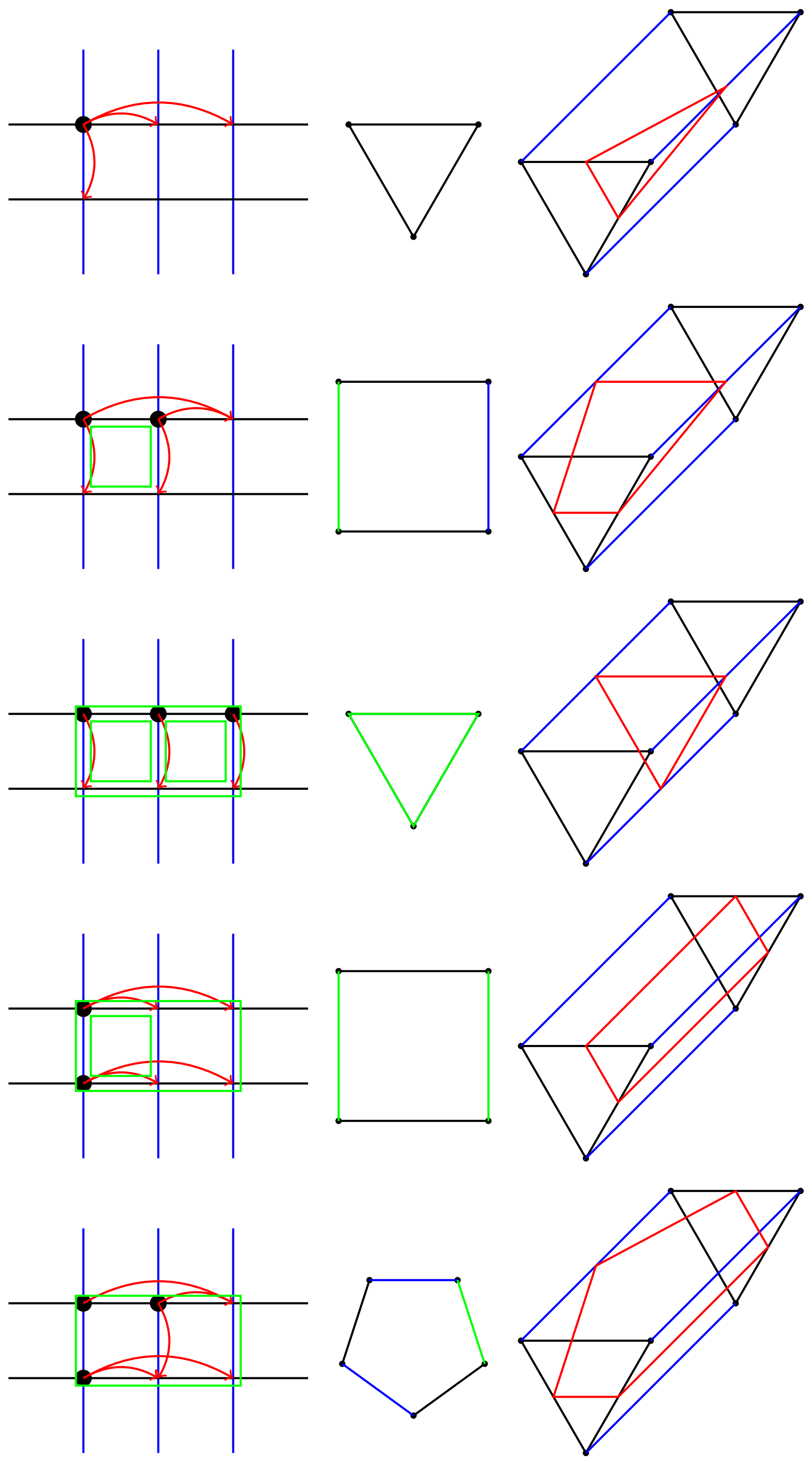


Figure 6.7: Sections of the triangular prism, found using cross-hatching. (Left) Cross-hatching diagrams. Red arrows indicate an intersection calculated, while green rectangles show which of these intersections are cousins. (Centre) Resulting section. Black lines indicate the two intersections share a father, blue a mother, and green that they are cousins. (Right) Corresponding plane through the prism.

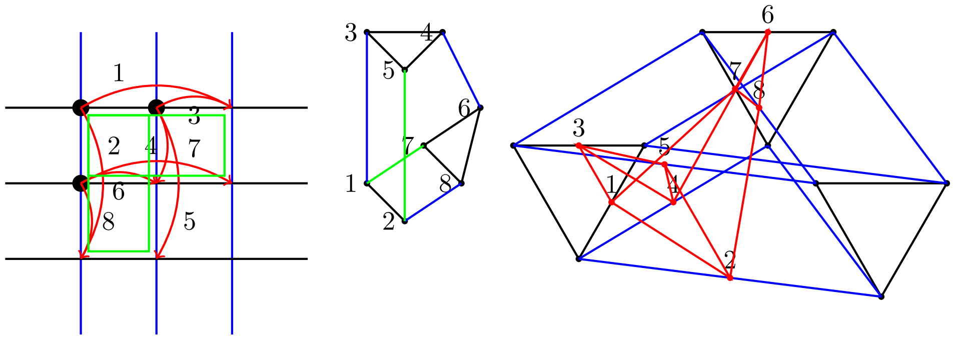


Figure 6.8: Example of a second order simplicial section in 5D. The section itself is a 3D polytope arising from the intersection of a 5-simplex with two 4D hyperplanes. It is called the 3-4 scutoid [23].

6.11 Conclusions

This chapter presents the final version of the intersection algorithm first presented in chapter 4. The algorithm now works in arbitrary dimensions and calculates the intersection of two simplices. Each step is proven to be self-consistent between all of its relevant calculations, as well as consistent with all relevant calculations of all previous steps. It is therefore robust.

Several practical considerations have been considered. These may be useful for other algorithms with similar concerns. Employing all of these provides greater efficiency to the algorithm.

Some techniques for visualizing these simplicial intersections are also presented. These give heuristic depictions of the first two layers of intersections, that is between a simplex and up to two hyperplanes.

6.12 Apocrypha

6.12.1 Adjacency cycles

In Section 6.9.2 I considered the problem of determining whether two sets J and K were adjacent. I resolved this by constructing the adjacency matrix under the combinadics representations of the sets. However, this was by no means the first attempt to solve this problem. In this section I present an earlier attempt that considers the graph of only those intersections that have been calculated.

Let $H(\Gamma)$ be the set of intersections $\mathbf{q}_\Gamma^J$ that all share the collection of hyperplanes Γ. Let $A(\Gamma)$ be the adjacency matrix of these intersections, using the definition of adjacency defined above. We then ask the question: Can we form $A(\Gamma \cup \{\eta\})$ from $H(\Gamma)$ and $A(\Gamma)$?

First note that the intersections in $H(\Gamma)$ fall into two groups: those with positive values in the $\mathbf{e}_\eta$ direction, $H_+(\Gamma)$ and; those with negative values in this direction, $H_-(\Gamma)$. The edges in $A(\Gamma)$ then fall into two groups as well: those between $H_+(\Gamma)$ and $H_-(\Gamma)$, $A_b(\Gamma)$, and; those within either group, $A_c(\Gamma)$. The graph $(H(\Gamma), A_b(\Gamma))$ is then bipartite.

For each edge in this bipartite graph we calculate an intersection of $H(\Gamma \cup \{\eta\})$. This intersection has the indices of the vertices of X of both its parents. Therefore,

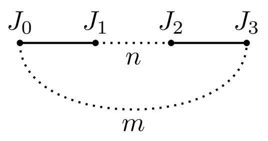


Figure 6.9: Configuration of four intersections of a given generation.

any other child of either parent, called a sibling, will be adjacent to this intersection. It remains to determine if any other intersections of this generation are adjacent.

Lemma 6.10. *Let a father be a given parent of a child node, and let the mother of this child be the other parent.* ***If the path length between any two nodes is equal to the number of indices by which they differ then*** *two children of a given generation of intersections are adjacent under one of three circumstances:*

siblings *the children share exactly one parent;*

cousins *the two fathers are adjacent as are the two mothers, i.e. the four parents form a 4-cycle;*

second cousins *the two fathers are adjacent while the two mothers are both adjacent to a fifth intersection, i.e. the four parents are part of a 5-cycle.*

Additionally, it is assumed that these cycles represent the shortest paths between each node of the cycle.

Proof Consider four nodes of $H(\Gamma)$, indexed by J_0, J_1, J_2 and J_3. Suppose J_0 and J_1 are adjacent, as are J_2 and J_3. These intersections may be configured as seen in Figure 6.9, where there are at least n edges between nodes J_1 and J_2 that do not pass through J_3 and at least m edges between J_0 and J_3 that do not pass through J_2. Without loss of generality $n \leq m$.

Suppose that the edge between J_0 and J_1 belongs to $A_b(\Gamma)$, as does the edge between J_2 and J_3. Then nodes of $H(\Gamma \cup \{\eta\})$ will be calculated for $J_0 \cup J_1$ and $J_2 \cup J_3$. These two nodes will be adjacent if $|(J_0 \cup J_1) \cap (J_2 \cup J_3)| = |J_0 \cup J_1| - 1$.

Note that if $0 < n = |J_1|$ then J_2 has no indices in common with J_1. Likewise, J_0 has no indices in common with J_3. The two nodes of the next generation therefore cannot be adjacent.

Since J_1 and J_2 are separated by n edges $|J_1 \cap J_2| = |J_1| - n$. The nodes J_0 and J_2 are separated by $n+1$ edges, and so $|J_0 \cap J_2| = |J_1| - (n+1)$. Therefore, J_2 contains $J_1 \setminus J_0$ but does not contain any of $J_0 \setminus J_1$ and $(J_0 \cup J_1) \cap J_2 = J_1 \cap J_2$.

Likewise, $|J_1 \cap J_3| = |J_1| - (n+1)$ and J_1 contains none of $J_3 \setminus J_2$. The nodes J_0 and J_3 are separated by the smaller of $n+2$ and m edges. Since $m \geq n$, this means that $|J_1| - (n+2) \leq |J_0 \cap J_3| \leq |J_1| - n$. We consider each of the three possibilities in turn.

If $|J_0 \cap J_3| = |J_1| - (n+2)$ then J_3 contains no indices of $J_0 \setminus J_1$ and $(J_0 \cup J_1) \cap J_3 = J_1 \cap J_3$. Moreover, $J_1 \cap (J_2 \cup J_3) = J_1 \cap J_2$ and $|(J_0 \cup J_1) \cap (J_2 \cup J_3)| = |J_1| - n = |J_0 \cup J_1| - n - 1$. The two children must then be siblings if they are to be adjacent.

If $|J_0 \cap J_3| = |J_1| - n$ then, by symmetry, J_3 contains $J_0 \setminus J_1$ but not $J_1 \setminus J_0$. Therefore, $J_3 \setminus J_2 = J_0 \setminus J_1$. Put another way, the transition from J_2 to J_3 is the inverse of the

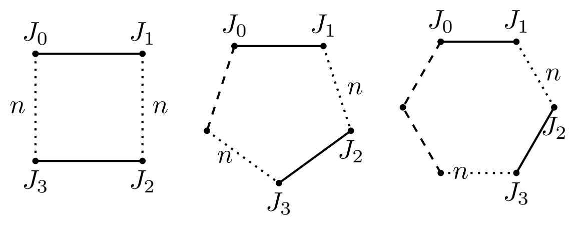


Figure 6.10: The three choices of m in Figure 6.9: $m = n$ (left); $m = n + 1$ (centre) and; $m \geq n + 2$ (right).

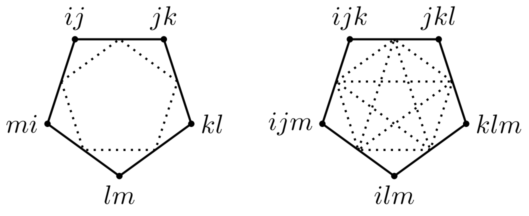


Figure 6.11: The two possible cases of second cousins. Left: only siblings are adjacent. Right: all children are adjacent.

transition from J_0 to J_1. This means that

$$\begin{aligned}(J_0 \cup J_1) \cap (J_2 \cup J_3) =&((J_0 \setminus J_1) \cup J_1) \cap (J_2 \cup (J_0 \setminus J_1))\\ =&((J_0 \setminus J_1) \cap J_2) \cup (J_1 \cap J_2) \cup (J_1 \cap (J_0 \setminus J_1)) \cup (J_0 \setminus J_1)\\ =&(J_1 \cap J_2) \cup (J_0 \setminus J_1)\end{aligned}$$

which has cardinality $|J_1|-n+1 = |J_0 \cup J_1|-n$. The two children will then be adjacent for $n = 1$, which will be referred to as cousins.

If $|J_0 \cap J_3| = |J_1| - (n + 1)$ then J_3 contains as many indices of J_0 as it does J_1. This means that either J_3 contains neither $J_0 \setminus J_1$ nor $J_1 \setminus J_0$, or it contains both. The former case has cardinality identical to when $|J_0 \cap J_3| = |J_1| - (n+2)$, while the latter case to when it equals $|J_1| - n$. This latter case therefore allows two children to be adjacent for $n = 1$, which will be referred to as second cousins.

The configuration with $n = 1$ and $m = 2$ necessarily places the four nodes into a 5-cycle. The arrangements of vertex sets that adhere to this condition are limited. If the five vertex sets share all but four or more vertices then all five are adjacent. The same is true if they share all but one. The two cases are then when the five sets share all but two or three vertices.

Figure 6.11 shows these two scenarios. To construct them, one may begin at any node with two vertices. Travel along one edge to the second node by changing one of the vertices. Travel along the other edge off of the first node by changing a different vertex so that this third node is not adjacent to the second. For example, in the case where the vertex sets share all but two vertices one can start with the set $\{i, j\}$. The adjacent sets are then $\{j, k\}$ and $\{m, i\}$. The final two nodes are adjacent to one another as well as the second and third nodes, but not the first. This uniquely determines the sets.

As shown in Figure 6.11 the children of the case where all but two vertices are shared are adjacent only if they are siblings. Take $J_0 = \{i, j\}$ and $J_3 = \{l, m\}$. The set J_3 clearly does not contain i or k, the vertices of $J_0 \setminus J_1$ and $J_1 \setminus J_0$, respectively. The children $\{k, l, m\}$ and $\{i, j, k\}$ differ by two vertices and therefore are not adjacent.

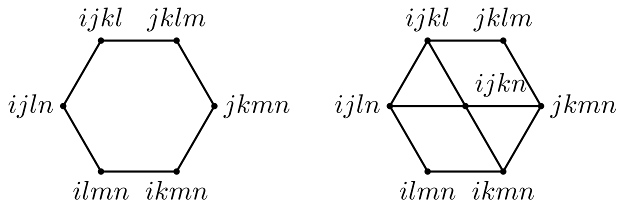


Figure 6.12: Counterexamples to necessary conditions for Lemma 6.10. (Left) The path length between any two nodes in this graph can be as high as 3, while any two nodes differ by at most 2 indices. (Right) An additional path passes through the cycle, making some of the nodes closer.

Meanwhile, all children are adjacent when the vertex sets share all but three vertices. Using $J_0 = \{i, j, k\}$ and $J_3 = \{i, l, m\}$ both i and l are within J_3. The children $\{i, k, l, m\}$ and $\{i, j, k, l\}$ differ by only one vertex and are adjacent. By symmetry the same is true of the other children around this cycle. ■

There are three barriers to using this lemma to construct $A(\Gamma \cup \{\eta\})$ from $A(\Gamma)$. Firstly, the case of second cousins requires knowledge of which vertices are shared between parents. Therefore, some comparison of vertex sets is required.

Secondly, it is not true in general that the path length between two nodes will equal the number of indices not shared. For example, consider the graph of vertex sets in the left of Figure 6.12. The path length between nodes $\{i, j, k, l\}$ and $\{i, k, m, n\}$ is three, but they differ by only two indices. Thus, path length does not indicate the number of common indices.

Thirdly, it is possible that there is a shorter path between any two nodes in a cycle that does not lie entirely on the cycle. The right of Figure 6.12 shows such a setup. The nodes $\{i, j, k, l\}$ and $\{i, l, m, m\}$ have a path length of two between them, as do $\{j, k, l, m\}$ and $\{i, k, m, n\}$. In the proof of Lemma 6.10 it is then assumed that the path length between $\{i, j, k, l\}$ and $\{i, k, m, n\}$ is three, but this second path reduces that number to two.

Ultimately, this means the number of graph configurations giving rise to adjacent children is too numerous to thoroughly test. Continuing in this manner will almost certainly become more computationally expensive than comparing all vertex sets of a given generation.

While this may be a dead end, for the sake of completeness I provide an algorithm for constructing the adjacency matrix $A(\Gamma \cup \{\eta\})$ considering only those circumstances of Lemma 6.10.

Algorithm 6.3. *First, construct $B(\Gamma \cup \{\eta\})$, the intergenerational adjacency matrix. Every time a child node is found, place two 1's in its respective column of $B(\Gamma \cup \{\eta\})$ corresponding to the two rows of its parent.*

Consider a particular child node and its row of $A(\Gamma \cup \{\eta\})$. To find its siblings, take the 'xor' combination of the two rows of $B(\Gamma \cup \{\eta\})$ associated with the child's parents.

To find its cousins, consider its father node J_f and mother node J_m. For every node K_f adjacent to the father, except J_m, check if it is a father node with a child. Take the mother K_m of the child of K_f. If K_m and J_m are adjacent then so are the two children.

To find second cousins, repeat the procedure for cousins, finding J_f, J_m, K_f and K_m. If K_m and J_m are found to not be adjacent, take the 'and' combination of the *rows of* $A(\Gamma)$ *associated with these sets. If this combination has at least one non-zero element then the two children may be adjacent, depending on the number of indices the vertex sets of the parents have in common. This step must also be done reversing the role of father and mother.*

Final remarks and future works

The chaotic and cyclic ASPN sequences shows how problematic some nonlinearities can be when employing acceleration techniques, or equivalently when preconditioning certain methods with domain decomposition methods. Similar examples likely exist for other methods of this type, such as ASPIN, RASPEN and MSPIN. One can investigate if such counterexamples differ much between these types, and if so what this says about each type and their stability.

Ideally, I would like to determine what conditions can be placed on the nonlinearities to ensure convergence. From this research it is clear the shape of the fixed point iteration must have particular slope at all points. For example, a domain decomposition method that converges faster would have a fixed point iteration with a slope closer to zero, and thus farther from that of the dangerous contour lines. Brief changes in the slope could still create cycles after acceleration, but these may have significantly smaller basins of attraction. However, accelerating methods that already converge quickly would not be as useful.

There exist many extrapolation and Krylov subspace methods not mentioned in chapter 3. Connecting these into the framework may give new methods of the other type. While such an extrapolation method may not be popular, its corresponding Krylov method may prove useful.

Now that the connection between extrapolation methods and Krylov subspace methods has been made explicit, the goal is to apply extrapolation methods to methods such as ASPN. That is, the extrapolation methods will be used to resolve the Newton-Raphson acceleration steps of the algorithm. Whether this adds stability remains to be seen, though as before it is unlikely to guarantee convergence.

The robust simplicial intersection algorithm in chapters 4, 5 and 6 can be improved upon in a number of practical ways, from finding the ideal method for calculating intersections to improving storage of the information. Also of interest is applying this algorithm to intersections of mixed dimension. While it is not possible to robustly intersect two simplices of different dimensions, one can intersect a single simplex of lower dimension with a mesh of simplices of a larger dimension.

In the discussion of the combinatorial number system and adjacent index sets, it is noted that it would be ideal to have a formula for $N(J \cup K)$ based on $N(J)$ and $N(K)$, bypassing the need for recalculating the former at each step of the algorithm. Based on preliminary work it appears this formula follows similar patterns to the adjacency matrix. Having this formula would make the adjacency matrix and the combinatorial number system exceedingly useful for algorithms that must consider several combinations of objects simultaneously.

In the course of visualizing the simplicial intersections I was led to wonder if it might be possible to categorize and enumerate the different faces of these intersections. Certainly, we showed that the first order intersections, between a simplex and a hyperplane, are necessarily duoprisms. It remains open whether all orders of these intersections can be likewise easily identified, and if their flags can then uniquely identify the simplicial intersection as a whole. This last question has proven difficult even for the intersection of tetrahedra, but the first may still yield interesting results.

Bibliography

[1] Donald G. Anderson. Iterative procedures for nonlinear integral equations. *Journal of the Association for Computing Machinery*, 12:547–560, 10 1965.

[2] J. G. P. Barnes. An algorithm for solving non-linear equations based on the secant method. *The Computer Journal*, 8:66–72, 4 1965.

[3] Franklin H. Branin. Widely convergent method for finding multiple solutions of simultaneous nonlinear equations. *IBM Journal of Research and Development*, 16(5):504–522, 1972.

[4] R.P. Brent. On the davidenko-branin method for solving simultaneous nonlinear equations. *IBM Journal of Research and Development*, 16(4):434–436, 1972.

[5] C. Brezinski. Numerical stability of a quadratic method for solving systems of non linear equations. *Computing*, 14:205–211, 1975.

[6] Claude Brezinski. Accélération de la convergence en analyse numérique, 1977.

[7] C. G. Broyden. A class of methods for solving nonlinear simultaneous equations. *Mathematics of Computation*, 19:577–593, 1965.

[8] S. Cabay and L. W. Jackson. A polynomial extrapolation method for finding limits and antilimits of vector sequences. *SIAM Journal on Numerical Analysis*, 13:734–752, 10 1976.

[9] Xiao-Chuan Cai and David E. Keyes. Nonlinearly preconditioned inexact Newton algorithms. *SIAM Journal on Scientific Computing*, 24(1):183–200, 2002.

[10] Rosetta Code. Sutherland-hodgman polygon clipping, 2021.

[11] Mike Cyrus and Jay Beck. Generalized two-and three-dimensional clipping. *Computers & Graphics*, 3(1):23–28, 1978.

[12] D.F. Davidenko. On a new method of numerical solution of systems of nonlinear equations. In *Dokl. Akad. Nauk SSSR*, volume 88, pages 601–602, 1953.

[13] V. Dolean, M. J. Gander, W. Kheriji, F. Kwok, and R. Masson. Nonlinear preconditioning: How to use a nonlinear Schwarz method to precondition Newton's method. *SIAM Journal on Scientific Computing*, 38:3357–3380, 2016.

[14] R.P. Eddy. Extrapolating to the limit of a vector sequence. *Information Linkage Between Applied Mathematics and Industry*, pages 387–396, 1 1979.

[15] V. Eyert. A comparative study on methods for convergence acceleration of iterative vector sequences. *Journal of Computational Physics*, 124:271–285, 1996.

[16] Haw Ren Fang and Yousef Saad. Two classes of multisecant methods for nonlinear acceleration. *Numerical Linear Algebra with Applications*, 16:197–221, 3 2009.

[17] Steven Fortune. Stable maintenance of point set triangulations in two dimensions. In *Annual Symposium on Foundations of Computer Science (Proceedings)*, pages 494–499, Los Alamitos, CA, USA, 10 1989. IEEE Computer Society.

[18] Martin J. Gander. On the origins of linear and non-linear preconditioning. In *Domain Decomposition Methods in Science and Engineering XXIII*, pages 153–161. Springer, 2017.

[19] Martin J. Gander and Caroline Japhet. An algorithm for non-matching grid projections with linear complexity. In *Domain decomposition methods in science and engineering XVIII*, pages 185–192. Springer, Berlin, Heidelberg, 2009.

[20] Martin J. Gander and Caroline Japhet. Algorithm 932: PANG: software for non-matching grid projections in 2D and 3D with linear complexity. *ACM Transactions on Mathematical Software*, 40(1):6, 2013.

[21] Walter Gander, Martin J. Gander, and Felix Kwok. *Scientific computing-An introduction using Maple and MATLAB*, volume 11. Springer Science & Business, 2014.

[22] E. Gekeler. On the solution of systems of equations by the epsilon algorithm of Wynn. *Mathematics of Computation*, 26, 1972.

[23] Pedro Gómez-Gálvez, Pablo Vicente-Munuera, Antonio Tagua, Cristina Forja, Ana M Castro, Marta Letrán, Andrea Valencia-Expósito, Clara Grima, Marina Bermúdez-Gallardo, Óscar Serrano-Pérez-Higueras, et al. Scutoids are a geometrical solution to three-dimensional packing of epithelia. *Nature communications*, 9(1):1–14, 2018.

[24] W. B. Gragg and G. W. Stewart. A stable variant of the secant method for solving nonlinear equations. *SIAM Journal on Numerical Analysis*, 13:889–903, 12 1976.

[25] Günther Greiner and Kai Hormann. Efficient clipping of arbitrary polygons. *ACM Transactions on Graphics*, 17(2):71–83, 1998.

[26] Christoph M. Hoffmann. The problems of accuracy and robustness in geometric computation. *Computer*, 22(3):31–39, 1989.

[27] K. Jbilou and H. Sadok. Vector extrapolation methods. Applications and numerical comparison. *Journal of Computational and Applied Mathematics*, 122:149–165, 10 2000.

[28] You-Dong Liang and Brian A. Barsky. A new concept and method for line clipping. *ACM Transactions on Graphics*, 3(1):1–22, January 1984.

[29] Lulu Liu and David E. Keyes. Field-split preconditioned inexact Newton algorithms. *SIAM Journal on Scientific Computing*, 37:A1388–A1409, 2015.

[30] S. Lui. On Schwarz alternating methods for nonlinear elliptic pdes. *SIAM Journal on Scientific Computing*, 21(4):1506–1523, 1999.

[31] Conor McCoid and Martin J. Gander. A provably robust algorithm for triangle-triangle intersections in floating-point arithmetic. *ACM Transactions on Mathematical Software*, 48(2):1–30, jun 2022.

[32] M. Mešina. Convergence acceleration for the iterative solution of the equations x = ax + f. *Computer Methods in Applied Mechanics and Engineering*, 10:165–173, 1977.

[33] William M. Newman and Robert F. Sproull. *Principles of Interactive Computer Graphics*. McGraw-Hill, Inc., New York, 1979.

[34] Ari Rappoport. An efficient algorithm for line and polygon clipping. *The Visual Computer*, 7(1):19–28, 1991.

[35] Youcef Saad and Martin H. Schultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 7, 7 1986.

[36] Jonathan Richard Shewchuk. Adaptive precision floating-point arithmetic and fast robust geometric predicates. *Discrete & Computational Geometry*, 18(3):305–363, 1997.

[37] Avram Sidi. Extrapolation vs. projection methods for linear systems of equations. *Journal of Computational and Applied Mathematics*, 22:71–88, 1988.

[38] Avram Sidi. Efficient implementation of minimal polynomial and reduced rank extrapolation methods. *Journal of Computational and Applied Mathematics*, 36:305, 1991.

[39] Avram Sidi, William F. Ford, and David A. Smith. Acceleration of convergence of vector sequences. *SIAM Journal on Numerical Analysis*, 23:178–196, 1986.

[40] Václav Skala. An efficient algorithm for line clipping by convex polygon. *Computers & Graphics*, 17(4):417–421, 1993.

[41] David A. Smith, William F. Ford, and Avram Sidi. Extrapolation methods for vector sequences. *SIAM Review*, 29:199–233, 1987.

[42] Ivan E. Sutherland and Gary W. Hodgman. Reentrant polygon clipping. *Communications of the ACM*, 17(1):32–42, 1974.

[43] David Vanderbilt and Steven G. Louie. Total energies of diamond (111) surface reconstructions by a linear combination of atomic orbitals method. *Physical Review B*, 30:6118–6130, 11 1984.

[44] Homer F. Walker and Peng Ni. Anderson acceleration for fixed-point iterations. *SIAM Journal on Numerical Analysis*, 49:1715–1735, 2011.

[45] David S. Watkins. *Fundamentals of matrix computations*, volume 64. John Wiley & Sons, New York, 8 2004.

[46] Kevin Weiler and Peter Atherton. Hidden surface removal using polygon area sorting. In *Computer Graphics*, volume 11, pages 214–222, New York, 1977. ACM.

[47] Philip Wolfe. The secant method for simultaneous nonlinear equations. *Communications of the ACM*, 2:12–13, 1959.

[48] www.nba-live.com. NBA 2K13 screenshot, April 2013. [Date accessed: 10 May 2022].

[49] P. Wynn. On a device for computing the em(Sn) transformation. *Mathematical Tables and Other Aids to Computation*, 10:91–96, 1956.