

Boîte à outil matlab pour le calcul géométrique

D. Legland

15 juin 2007

Résumé

Descriptif des fonctions que j'ai développées sous matlab pour regrouper les calculs faits sur les objets géométriques (points, droites, cercles, polygones, courbes polynomiales...). Je précise les structures de données utilisées, et j'explique quelques fonctions.

Table des matières

1	Introduction	3
2	Considérations informatiques	4
2.1	Représentation des données	4
2.2	Gestion des cas particuliers	4
2.3	Gestion des erreurs d'arrondi	5
2.4	Primitives utiles	5
3	Points et vecteurs	6
3.1	Représentation	6
3.2	Création de points	6
3.3	Calculs sur les points	7
3.4	Manipulation de vecteurs	8
4	Droites, segments et demi-droites	9
4.1	Représentations	9
4.2	Fonctions de création	9
4.3	Mesures sur les droites	12
4.4	Opérations géométriques	12
4.5	Interactions entre droites et points	14
4.6	Tests de position	15
5	Polygones et polylignes	16
5.1	Représentation	16
5.2	Fonctions sur les polylignes	16
5.3	Mesures sur les polygones	18
5.4	Création et composition de polygones	19
5.5	Opérations sur les polygones	21

6	Cercles, ellipses, et autres coniques	22
6.1	Représentation	22
6.2	Fonctions de création	22
6.3	Fonctions de conversion	23
6.4	Fonctions de tests	24
7	Courbes polynomiales	25
7.1	Représentation	25
7.2	Fonctions de base	25
7.3	Calculs de courbures	26
8	Autres formes géométriques	28
8.1	Grilles de points	28
8.2	Mosaïques	28
9	Transformations affines	30
9.1	Représentation	30
9.2	Fonctions de création	30
9.3	Transformation de formes	31
10	Fonctions d’affichage	33
10.1	Objets génériques	33
10.2	Objets droits	33
10.3	Polygones	34
10.4	Objets courbes	34

Chapitre 1

Introduction

L'idée de cette bibliothèque est de fournir un ensemble de fonctions permettant les calculs sur des primitives géométriques courantes. Les primitives géométriques considérées sont par exemple les points, les droites, les cercles ou les polygones, et les calculs proposés sont par exemple des calculs d'intersections (entre deux droites, entre une droite et un cercle...), de transformation, ou de mesure (aire et périmètre d'un polygone, longueur d'une courbe polynomiale...).

Les fonctions ont été écrites dans le but de cacher la partie calculatoire, en essayant de fournir des noms de fonctions suffisamment explicites pour pouvoir trouver rapidement la fonction nécessaire à une tâche donnée. La bibliothèque ne prétend sûrement pas être complète, mais essaie plutôt de fournir une base stable, qui permet de d'implémenter rapidement des algorithmes plus évolués sans avoir à réinventer sans cesse la roue.

Dans ce document, les fonctions sont classées en fonctions des primitives acceptées en argument (fonctions pour les points, fonctions pour les droites...). Cette distinction n'existe pas pour l'implémentation, toutes les fonctions étant stockées dans le même répertoire.

Chapitre 2

Considérations informatiques

2.1 Représentation des données

On essaie de représenter chaque primitive géométrique par un ensemble de valeurs numériques. On souhaite qu'une série de valeurs puisse définir de manière unique une primitive. Par contre l'inverse n'est pas nécessaire : une même primitive peut être définie par plusieurs séries de valeurs. Par exemple une droite peut être définie par plusieurs couples de points.

En général, tous les paramètres définissant une figure peuvent être placés dans un seul vecteur ligne. Cela permet de considérer un ensemble de figures en utilisant simplement un tableau de valeurs.

On tente d'éviter dans la mesure du possible les cas pathologiques pour l'ordinateur (on évite de considérer la pente d'une droite verticale ou presque verticale, qui tend vers l'infini).

2.2 Gestion des cas particuliers

Le résultat de certaines fonctions peut ne pas être défini dans certains cas particuliers, par exemple le point d'intersection de deux droites parallèles, ou le cercle passant par 3 points alignés.

La convention utilisée est de retourner un vecteur ne contenant que des valeurs NaN lorsqu'une telle situation se présente.

Pour certains cas, on considère aussi les valeurs infinies. Par exemple, le point d'intersection de deux droites confondues est donné par les coordonnées $[\infty, \infty]$.

2.3 Gestion des erreurs d'arrondi

Les nombres manipulés ayant avec une précision finie, les calculs peuvent ne pas tomber juste. Par exemple, le calcul de la distance entre un point et une droite peut renvoyer un résultat non nul même si le point appartient à la droite.

La solution retenue est d'utiliser une valeur de tolérance. Deux points sont donc égaux si leur distance est inférieure ou égale à la valeur de tolérance. En pratique, une valeur de tolérance de 10^{-14} est utilisée.

Une amélioration appréciable des fonctions consisterait à pouvoir spécifier la tolérance de manière systématique.

2.4 Primitives utiles

2.4.1 Angles

De manière générale, la bibliothèque utilise des angles en radians, orientés positivement dans le sens trigonométrique (inverse des aiguilles d'une montre), et calibrés entre 0 et 2π .

2.4.2 Boîte

La boîte est un rectangle dont les bords sont parallèles aux axes considérés. Cette structure de donnée est surtout utilisée pour définir les coordonnées minimales et maximales d'une primitive, ou bien pour sélectionner les primitives dans une zone donnée.

Une boîte est définie par les 4 valeurs `[xmin, xmax, ymin, ymax]`.

Chapitre 3

Points et vecteurs

3.1 Représentation

Un point (ainsi qu'un vecteur) est représenté par un vecteur ligne contenant ses coordonnées cartésiennes x et y . En générale, la bibliothèque ne fait de différence formelle entre un point et un vecteur.

Un ensemble de points est représenté par une matrice $n \times 2$, où chaque ligne représente un point.

3.2 Création de points

3.2.1 centroid

Calcule le centroïde, éventuellement pondéré, d'un ensemble de points.

entrée un ensemble de points, donnés soit en arguments séparés, soit par un tableau $n \times 2$, et éventuellement une pondération pour chaque point.

sortie le centroïde des points donnés en argument.

3.2.2 polarPoint

Calcule les coordonnées d'un point donné en coordonnées polaires (angle+distance à l'origine), en précisant éventuellement un point de référence.

entrée le point de référence (optionnel), les coordonnées polaires du point.

sortie les coordonnées cartésiennes du point créé.

3.3 Calculs sur les points

3.3.1 angle2points

Calcule l'angle par rapport à l'horizontal formé par la droite passant par deux points. L'angle est compté en radians, positif dans le sens trigonométrique, et calibré entre 0 et 2π .

entrée deux points.

sortie l'angle avec l'horizontale de la droite passant par les deux points.

Si les entrées sont deux tableaux de n points, le résultat est un vecteur colonne de n valeurs représentant les angles correspondants à chaque couple de points.

3.3.2 angle3Points

Calcule l'angle formé par 3 points, le deuxième point étant pris comme point de mesure de l'angle. Le résultat est calibré entre 0 et 2π .

entrée trois points.

sortie l'angle entre les 3 points.

3.3.3 angleSort

Trie un ensemble de points en fonction de leur position angulaire par rapport à l'origine ou par rapport à un point de référence. Le polygone formé par les points après le tri ne se recoupe pas.

entrées un point de référence (optionnel), un ensemble de points.

sortie les même points qu'en entrée, triés selon leur position angulaire, et éventuellement l'ordre de tri.

3.3.4 distancePoints

Calcule la distance entre deux points, entre un points et un groupe de points, ou entre deux groupes de points.

entrée un point ou un groupe de points, un point ou un groupe de points, et éventuellement la métrique utilisée.

sortie la distance entre les deux points, ou un vecteur colonne contenant les distances entre le point isolé et chaque point du groupe de points, ou alors les distances calculées entre chaque couple de points ayant le mêmes indices dans les deux groupes.

3.3.5 minDistancePoints

Calcule la plus petite distance entre des points. Selon la dimension des arguments, plusieurs cas peuvent être considérés : distance minimale entre un point et un groupe de points, distance minimale entre tous les couples de points possibles parmi un groupe, ou distance minimale calculée pour chaque point d'un groupe par rapport à un autre groupe.

entrée les coordonnées des points (un ou deux groupes de points), et éventuellement la métrique utilisée.

sortie la distance minimale (ou un vecteur colonne contenant les distances minimales calculées pour chaque point du premier groupe) au deuxième groupe de points.

3.4 Manipulation de vecteurs

3.4.1 vecnorm

Calcule la norme d'un vecteur.

entrée les coordonnées du vecteur, ou un groupe de coordonnées.

sortie la norme du vecteur, ou un ensemble de normes.

3.4.2 normalize

Calcule le vecteur normalisé, c'est à dire tel que sa norme vaut 1.

entrée un vecteur ou un ensemble de vecteurs.

sortie les vecteurs normalisés, avec une norme égale à 1.

Chapitre 4

Droites, segments et demi-droites

4.1 Représentations

On considère plusieurs types d'objets inscrits dans une droite : les droites proprement dites (line), les segments (edge), les demi droites (ray), et les arcs de droites (linearc).

Pour les droites et les demi-droites, on considère 4 paramètres : les coordonnées d'un point origine (x_0, y_0) , et les valeurs d'un vecteur directeur (d_x, d_y) , concaténés dans un vecteur ligne de 4 éléments : $[x_0, y_0, d_x, d_y]$.

Pour les segments, on considère les coordonnées du premier point et les coordonnées du deuxième point, concaténées dans un vecteur ligne de 4 éléments : $[x_1, y_1, x_2, y_2]$.

Pour un arc de droite, on considère 6 paramètres : les 4 paramètres de la droite support, et 2 paramètres t_0 et t_1 marquant les limites de paramétrisation de l'objet : $[x_0, y_0, d_x, d_y, t_0, t_1]$. On retrouve une droite avec $t_0 = -\infty$ et $t_1 = \infty$, une demi-droite si $t_1 = \infty$, et un segment de droite si $t_0 = 0$ et $t_1 = 1$.

4.2 Fonctions de création

Ces fonctions permettent de créer simplement les structures de données correspondant aux primitives manipulées.

4.2.1 createLine

Crée une droite à partir de deux points.

entrée les coordonnées de deux points.

sortie une paramétrisation de la droite passant par les deux points. L'origine de la droite est le premier point, le vecteur directeur est le vecteur joignant les deux points.

4.2.2 parallelLine

Crée une droite parallèle à une droite donnée.

entrée la paramétrisation d'une droite, et les coordonnées d'un point.

sortie une droite parallèle à la droite donnée en argument, et passant par le point spécifié.

4.2.3 orthogonalLine

Crée une droite perpendiculaire à une droite donnée.

entrée la paramétrisation d'une droite, et les coordonnées d'un point.

sortie une droite perpendiculaire à la droite donnée en argument, et passant par le point spécifié.

4.2.4 medianLine

Crée la médiatrice de deux points.

entrée les coordonnées de deux points.

sortie une paramétrisation de la médiatrice des deux points. L'origine de la droite est le centre des points, le vecteur directeur est le vecteur joignant les deux points augmenté de $\pi/2$.

4.2.5 cartesianLine

Crée une droite à partir de ses coordonnées cartésiennes : $ax + by + c = 0$.

entrée les trois paramètres a , b et c .

sortie une paramétrisation de la droite vérifiant $ax + by + c = 0$.

4.2.6 bisector

Crée la bissectrice de 3 points ou de 2 droites. Dans le cas où trois points A , B et C sont passés en argument, la fonction renvoie la bissectrice des deux droites (BA) et (BC) .

entrée les coordonnées des trois points, ou les paramétrisations des deux droites.

sortie la bissectrice des points ou des droites.

4.2.7 invertLine

Inverse l'orientation d'une droite. Le changement d'orientation change la paramétrisation de la droite, mais le tracé de la droite reste le même.

entrée la paramétrisation d'une droite.

sortie la droite avec le même point origine mais avec le vecteur directeur opposé.

4.2.8 createEdge

Crée un segment de droite à partir de deux points. L'intérêt de cette fonction est d'une part de cacher à l'utilisateur la représentation interne de la paramétrisation, d'autre part de rendre le code plus explicite.

entrée les coordonnées de deux points.

sortie la paramétrisation du segment.

4.2.9 lineFit

Ajuste une droite à un ensemble de points, au sens des moindres carrés de la distance euclidienne.

entrée un ensemble de points.

sortie une paramétrisation de la droite, telle que la somme des distances au carré entre la droite et chaque point soit minimale.

4.3 Mesures sur les droites

4.3.1 lineAngle

Calcule l'angle d'une droite avec l'axe horizontal, ou l'angle entre deux droites.

entrée la paramétrisation de la (ou des) droites.

sortie l'angle entre les deux droites, ou l'angle avec l'horizontale, calibré entre 0 et 2π .

4.3.2 edgeAngle

Calcule l'angle d'un segment de droite avec l'axe horizontal.

entrée la paramétrisation du segment.

sortie l'angle avec l'horizontale, calibré entre 0 et 2π .

4.3.3 edgeLength

Calcule la longueur d'un segment de droite.

entrée la paramétrisation du segment.

sortie la longueur du segment.

4.4 Opérations géométriques

4.4.1 intersectLines

Calcule le point d'intersection de deux droites.

entrée la paramétrisation des deux droites.

sortie les coordonnées du point d'intersection. Si les droites sont parallèles, renvoie `[NaN, NaN]`. Si les droites sont confondues, renvoie `[∞ , ∞]`.

4.4.2 intersectEdges

Calcule le point d'intersection de deux segments de droite.

entrée la paramétrisation des deux segments.

sortie les coordonnées du point d'intersection. Si les segments sont parallèles, renvoie $[\text{NaN}, \text{NaN}]$. Si les segments sont colinéaires, renvoie $[\infty, \infty]$.

4.4.3 intersectLineEdge

Calcule le point d'intersection d'un segment et d'une droite.

entrée la paramétrisation de la droite, et la paramétrisation du segment.

sortie les coordonnées du point d'intersection. Si les objets sont parallèles, renvoie $[\text{NaN}, \text{NaN}]$. Si les objets sont colinéaires, renvoie $[\infty, \infty]$.

4.4.4 clipLineRect

Calcule l'intersection d'une droite avec une boîte.

entrée la paramétrisation de la droite, et les coordonnées extrêmes de la boîte.

sortie les coordonnées du segment de droite contenu dans la boîte. Si la droite ne coupe pas le rectangle, renvoie un vecteur ligne de 4 éléments ne contenant que des valeurs NaN.

4.4.5 clipEdge

Calcule l'intersection d'un segment de droite avec une boîte.

entrée la paramétrisation du segment, et les coordonnées extrêmes de la boîte.

sortie les coordonnées du segment de droite contenu dans la boîte. Si le segment ne coupe pas le rectangle, renvoie un vecteur ligne de 4 éléments ne contenant que des valeurs NaN.

4.5 Interactions entre droites et points

4.5.1 distancePointLine

Calcule la plus courte distance entre un point et une droite.

entrée les coordonnées du point, et la représentation de la droite.

sortie la distance entre le point et la droite.

4.5.2 distancePointEdge

Calcule la plus courte distance entre un point et un segment.

entrée les coordonnées du point, et la représentation du segment.

sortie la distance entre le point et le segment.

4.5.3 linePosition

Calcule la position t_p d'un point sur une droite, telle que les coordonnées du point soit égale à $(x_0 + t_p d_x, y_0 + t_p d_y)$.

entrée les coordonnées du point (ou d'un groupe de points), la représentation de la droite.

sortie la position du point sur la droite (ou un vecteur colonne de positions).

4.5.4 pointOnLine

Crée un point sur une droite, à une distance donnée de l'origine de la droite.

entrée une droite, et une distance (signée).

sortie les coordonnées du point créé.

4.5.5 projPointOnLine

Projette un point sur une droite.

entrée les coordonnées du point à projeter, et la paramétrisation de la droite.

sortie les coordonnées du point projeté.

4.6 Tests de position

Ces différentes fonctions permettent de localiser un point par rapport à un objet ligne, segment, ou demi-droite. Les conventions de nommage ne sont pas respectées, car du fait qu'elles renvoient une valeur booléenne, il serait plus logique que le nom commence par « isXXX ».

4.6.1 onLine

Teste si un point appartient à une droite.

entrée les coordonnées du point, les coordonnées de la droite.

sortie renvoie 1 si le point est sur la droite, 0sinon.

4.6.2 onEdge

Teste si un point appartient à un segment de droite.

entrée les coordonnées du point, les coordonnées du segment.

sortie renvoie 1 si le point est sur le segment, 0sinon.

4.6.3 onRay

Teste si un point est sur une demi-droite.

entrée les coordonnées du point, les coordonnées de la demi-droite.

sortie renvoie 1 si le point est sur la demi-droite, 0sinon.

4.6.4 isLeftOriented

Teste si un point est localisé « à gauche » d'une droite. On considère qu'il est à gauche par rapport à la direction du vecteur de la droite.

entrée les coordonnées du point, la paramétrisation de la droite.

sortie renvoie 1 si le point est du côté gauche de la droite, 0 sinon.

Chapitre 5

Polygones et polylignes

On distingue le polygone, qui est une figure fermée, qui ne se croise pas, et la polyligne, qui est un ensemble de segments de droites connectés les uns aux autres, et qui peut éventuellement se croiser.

5.1 Représentation

Les deux primitives sont définies à partir d'une série de points, avec en plus pour le polygone l'hypothèse que le dernier point est relié au premier point.

Pour les polygones complexes (polygones disjoints, ou polygones troués), on considère un ensemble de polygones simples. Chaque polygone est orienté. Pour le stockage, on considère que deux polygones simples sont séparés par un « Not-a-point » avec les coordonnées (NaN, NaN). Cependant, les fonctions ne considèrent en général que les polygones simples.

5.2 Fonctions sur les polylignes

Il s'agit ici de courbe représentées par un ensemble de points. Pour des mesures plus précises sur des courbes polynomiales, on peut consulter la section 7 qui leur est consacrée.

5.2.1 `curveLength`

Calcule la longueur de la courbe en additionnant la longueur des segments élémentaires.

5.2.2 `curveCentroid`

Calcule le centroïde de la courbe, en calculant le centroïde des milieux des segments, pondérés par la longueur des segments correspondants.

5.2.3 `parametrize`

Calcule une paramétrisation approximative d'une série de points. La paramétrisation correspond à la somme cumulée des longueurs des segments élémentaires.

5.2.4 `curvature`

[obsolète] Calcule une courbure approchée d'une courbe constituée d'un ensemble de points. Cette fonction commence par calculer une approximation polynomiale de la courbe, il est préférable d'utiliser la fonction `polynomialCurveCurvature`.

5.2.5 `cart2geod`

Convertit un point en coordonnées cartésiennes en un point en coordonnées « géodésiques », composées de la position géodésique sur la courbe, et la distance signée à la courbe.

5.2.6 `geod2cart`

Convertit un point donné par sa position géodésique et sa distance signée à une courbe (polyligne) en un point en coordonnées cartésiennes. L'intérêt de cette fonction est de créer rapidement une courbe parallèle à une autre.

5.3 Mesures sur les polygones

5.3.1 `polygonCentroid`

Calcule le centre de gravité d'un polygone.

entrée les coordonnées des sommets du polygone.

sortie les coordonnées du centre de gravité (centroïde géométrique).

5.3.2 `polygonArea`

Calcule l'aire signée d'une polygone. Le résultat est un nombre qui peut être négatif. Le signe du résultat dépend du sens de parcours des sommets du polygone. Si les sommets sont parcourus avec l'intérieur du polygone à gauche du contour, le résultat est positif. Sinon, il est négatif.

entrée les coordonnées des sommets du polygone.

sortie l'aire signée du polygone.

5.3.3 `polygonLength`

Calcule le périmètre d'un polygone.

entrée les coordonnées des sommets du polygone.

sortie le périmètre du polygone.

5.3.4 `polygonNormalAngle`

Calcule l'angle normal en un ou plusieurs sommet(s) d'un polygone.

entrée les coordonnées des sommets d'un polygone, et les indices des sommets pour lesquels on souhaite calculer l'angle normal.

sortie un vecteur colonne contenant l'angle normal pour chaque sommet spécifié.

Si on calcule les angles normaux de tous les sommets d'un polygone, leur somme vaut 2π .

5.3.5 supportFunction

Calcule la fonction support d'un polygone convexe. La fonction support est un outil de géométrie souvent utilisé dans les articles de géométrie stochastique, qui correspond à la plus petite distance à l'origine possible d'une droite normale à la direction considérée et contenant la figure étudiée. On en déduit une « signature » pour la figure, et certaines propriétés géométrique (diamètre de Feret, épaisseur moyenne...).

entrée les coordonnées des sommets d'un polygone convexe, et éventuellement le nombre d'angles à considérer, ou une série de valeurs angulaires.

sortie la valeur de la fonction support pour chaque valeur angulaire spécifiée.

5.3.6 convexification

Calcule la convexification d'une fonction support. C'est la fonction duale de supportFunction.

entrée une fonction support

sortie le polygone qui a même fonction support que celle spécifiée.

5.3.7 steinerPoint

Calcule le point de Steiner d'un polygone. Le point de Steiner est un centroïde calculé sur les sommets d'un polygone, mais pondéré par l'angle du polygone au sommet considéré.

entrée les coordonnées des sommets du polygone.

sortie les coordonnées du point de Steiner.

5.4 Création et composition de polygones

5.4.1 readPolygon

Charge une série de polygones depuis un fichier texte. Le fichier est constitué de plusieurs couples de lignes contenant le même nombre de valeurs numérique. Chaque ligne est une série de coordonnées, deux lignes définissent ainsi une série de point. Chaque couple de ligne définit un polygone simple.

entrée le nom du fichier.

sortie un tableau de cellules contenant un polygone par cellule.

5.4.2 rectAsPolygon

Convertit un rectangle donné par un coin, ses dimensions, et une orientation, en un polygone, représenté par 4 sommets.

entrée les paramètres du rectangle : coordonnées du coin inférieur gauche, largeur, hauteur, et orientation (égale à zéro par défaut).

sortie les coordonnées des coins du rectangle.

5.4.3 clipPolygon/polygonClipBox

Découpe un polygone à l'aide d'une boîte. La version implémentée ne fonctionne que pour les polygones convexes. Le résultat est donc soit vide, soit un polygone convexe.

entrée les coordonnées des sommets d'un polygone, les coordonnées maximales de la boîte.

sortie les coordonnées des sommets du polygone découpé.

5.4.4 clipPolygonHP/polygonClipHP

Découpe un polygone à l'aide d'un demi plan. L'idée est de pouvoir utiliser cet algorithme pour chacun des côtés d'une boîte, et donc de pouvoir découper un polygone par une boîte.

entrée les coordonnées des sommets d'un polygone, la paramétrisation de la droite.

sortie les coordonnées des sommets du polygone découpé.

5.4.5 steinerPolygon

Calcule le polygone de Steiner d'un ensemble de vecteurs. Le polygone de Steiner est obtenu en étirant successivement par chaque vecteur le point origine. Le polygone obtenu est convexe, et a ses côtés de même direction et de même longueur que les vecteurs de départ.

entrée une liste de vecteurs.

sortie les coordonnées des sommets du polygone de Steiner correspondant.

5.5 Opérations sur les polygones

5.5.1 `intersectLinePolygon`

Calcule les points d'intersection entre une droite et un polygone.

entrée la paramétrisation de la droite, les coordonnées des sommets du polygone.

sortie les coordonnées des points d'intersection

5.5.2 `polygonExpand`

Dilate un polygone en décalant chacun de ses côtés d'une distance donnée, et en prolongeant les nouveaux côtés pour créer les nouveaux sommets. La distance peut être positive, ce qui agrandit le polygone, ou négative, ce qui le rétrécit.

Ce n'est pas une dilatation morphologique, pour laquelle le résultat est un polygone arrondi.

entrée les coordonnées des sommets du polygone, et la distance de dilatation.

sortie le polygone dilaté.

5.5.3 `medialAxisConvex`

Calcule l'axe médian d'un polygone convexe. L'axe médian est une définition du squelette d'un polygone. Il a la forme d'un arbre (au sens de graphe) qui relie les sommets du polygone, et pour lequel chaque arête appartient à la bissectrice de deux côtés.

entrée les coordonnées des sommets du polygone.

sortie un ensemble de points représentant les sommets de l'arbre (dont les sommets du polygone), et un ensemble d'arêtes constitués d'un couple d'indices référençant les sommets source et destination.

Chapitre 6

Cercles, ellipses, et autres coniques

6.1 Représentation

Un cercle est défini par un vecteur ligne contenant 3 éléments : le centre x , le centre y , le rayon r . De plus, si on considère les cercles orientés, on ajoute un quatrième paramètre valant 1 si le cercle est direct (l'intérieur est à gauche) ou 0 si le cercle est indirect (l'intérieur est à droite du cercle).

Pour un arc de cercle, on ajoute deux éléments correspondants à l'angle de début, et à l'angle de fin.

6.2 Fonctions de création

6.2.1 createCircle

Crée un cercle à partir de 2 ou 3 points. Si trois points sont passés en paramètres, le cercle créé passe par les trois points. Si deux points sont donnés en paramètres, Ils définissent le diamètre du cercle.

entrée les coordonnées des points.

sortie la paramétrisation du cercle.

6.2.2 createDirectedCircle

Crée un cercle à partir de 3 points, en spécifiant l'orientation du cercle.

entrée les coordonnées des points, l'orientation du cercle.

sortie la paramétrisation du cercle.

6.2.3 enclosingCircle

Trouve le plus petit cercle qui contient un ensemble de points.

entrée un ensemble de points.

sortie le plus petit cercle englobant tous les points.

6.3 Fonctions de conversion

6.3.1 circleAsPolygon

Convertit un cercle en polygone, éventuellement en spécifiant le nombre de sommets.

entrée la paramétrisation du cercle, et éventuellement le nombre de sommets du polygone.

sortie un polygone approchant le cercle.

6.3.2 ellipseAsPolygon

Convertit une ellipse en polygone, éventuellement en spécifiant le nombre de sommets.

entrée la paramétrisation de l'ellipse, et éventuellement le nombre de sommets du polygone.

sortie un polygone approchant l'ellipse.

6.3.3 circleArcAsCurve

Convertit un arc de cercle en courbe, éventuellement en spécifiant le nombre de points.

entrée la paramétrisation de l'arc de cercle, et éventuellement le nombre de points à utiliser pour l'approximation.

sortie une série de points approchant l'arc de cercle.

6.4 Fonctions de tests

6.4.1 onCircle

Teste si un point est situé sur un cercle.

6.4.2 inCircle

Teste si un point est situé à l'intérieur d'un cercle.

Chapitre 7

Courbes polynomiales

7.1 Représentation

Une courbe polynomiale du plan est définie par :

- un ensemble de coefficients polynomiaux pour les coordonnées x
- un ensemble de coefficients polynomiaux pour les coordonnées y
- un domaine de variation de la variable utilisée, que l'on note t

On considère que si le domaine de définition de t n'est pas précisé, il vaut $[0 : 1]$. Les deux ensembles de coefficients sont stockés dans deux vecteurs lignes. Cela permet de considérer des paramétrisation de degré différent sur les axes x et y .

Les coefficients sont stockés par degré croissant en commençant par le degré 0. Un vecteur de 6 éléments définit ainsi un polynôme de degré 5.

7.2 Fonctions de base

7.2.1 `polynomialCurveFit`

Calcule les coefficients polynomiaux qui ajustent au mieux un ensemble de points. L'ajustement des coefficients se fait séparément sur les différentes coordonnées.

7.2.2 `polyfit2`

Fonction simplifiée d'ajustement de courbe polynomiale à un ensemble de points.

7.2.3 `polynomialCurvePoint`

Calcule les coordonnées d'un point sur une courbe polynomiale en fonction de la position t .

7.2.4 `polynomialCurvePosition`

Calcule la position t d'un point sur une courbe polynomiale en fonction de sa position géodésique.

7.2.5 `polynomialCurveLength`

Calcule la longueur géodésique d'une courbe polynomiale, en intégrant le Jacobien sur tout le domaine de paramétrisation.

7.2.6 `polynomialCurveCentroid`

Calcule le centroïde d'une courbe polynomiale.

7.3 Calculs de courbures

7.3.1 `polynomialCurveDerivative`

Calcule le vecteur dérivé d'une courbe polynomiale pour les positions spécifiées.

7.3.2 `polynomialCurveNormal`

Calcule le vecteur normal d'une courbe polynomiale pour les positions spécifiées. Le vecteur normal est orthogonal au vecteur tangent.

7.3.3 polynomialCurveCurvature

Calcule la courbure d'une courbe polynomiale pour les positions spécifiées.

7.3.4 polynomialCurveCurvatures

Calcule les courbures principales d'une surface de révolution obtenue en utilisant comme courbe génératrice une courbe polynomiale.

7.3.5 polynomialDerivate

Fonction utilitaire qui calcule la dérivée d'un polynôme donné comme un vecteur ligne de coefficients.

Chapitre 8

Autres formes géométriques

8.1 Grilles de points

8.1.1 `squareGrid`

Genère un ensemble de sommets et d'arêtes disposés en carrés.

8.1.2 `triangleGrid`

Genère un ensemble de sommets et d'arêtes disposés en triangles.

8.1.3 `hexagonalGrid`

Genère un ensemble de sommets et d'arêtes disposés en hexagones.

8.2 Mosaïques

8.2.1 `crackPattern`

Genère des fissures droites à partir d'un ensemble de points, qui se propagent dans les directions spécifiées. Chaque fissure s'arrête lorsqu'elle rencontre une autre.

8.2.2 crackPattern2

Une variante du modèle de fissure : chaque point initie 3 fissures à 120 degrés les une des autres.

Chapitre 9

Transformations affines

9.1 Représentation

Une transformations affine est définie par une matrice 3×3 , dont la dernière ligne est égale à $(0, 0, 1)$:

$$T = \begin{bmatrix} m_{00} & m_{01} & m_{02} \\ m_{10} & m_{11} & m_{12} \\ 0 & 0 & 1 \end{bmatrix}$$

Les coefficients de la dernière colonne définissent le vecteur de translation. Les quatre coefficients du coin supérieur gauche définissent rotation, agrandissement et déformation. Si le déterminant de la matrice vaut 1, alors la transformation préserve les aires.

Une généralisation pourrait être de considérer les transformations homogènes, pour laquelle la dernière ligne n'est pas forcément égale à $[0, 0, 1]$.

9.2 Fonctions de création

9.2.1 translation

Crée une transformation affine représentant une translation.

entrée un vecteur de déplacement, donné soit par un vecteur ligne de 2 éléments, soit par deux arguments séparés.

sortie une matrice 3×3 représentant la translation.

9.2.2 rotation

Crée une transformation affine représentant une rotation d'un angle donné (en radians dans le sens trigonométrique), soit autour de l'origine, soit autour d'un point passé en argument.

9.2.3 scaling

Crée une transformation affine représentant un agrandissement. Si un seul nombre est donné en argument, le même agrandissement est appliqué selon les deux directions x et y . Si deux valeurs sont spécifiées, l'agrandissement est différent selon les axes.

9.2.4 lineSymmetry

Crée une transformation affine représentant une symétrie orthogonale par rapport à une droite.

9.2.5 homothecy

Crée une transformation affine représentant une homothétie, correspondant à un changement d'échelle uniforme dans toutes les directions autour d'un point spécifié.

9.3 Transformation de formes

9.3.1 transformPoint

Retourne les coordonnées du ou des points passés en paramètres.

9.3.2 transformVector

Renvoie les coordonnées d'un ou de plusieurs vecteurs transformés par une transformation affine.

9.3.3 transformLine

Renvoie les coordonnées d'une ou de plusieurs droites transformées par une transformation affine.

9.3.4 transformEdge

Renvoie les coordonnées d'un ou de plusieurs segments de droite transformés par une transformation affine.

Chapitre 10

Fonctions d’affichage

Ces fonctions dessinent la ou les figures spécifiées sur l’axe courant. Dans certains cas (droites, demi-droites...), la figure est découpée en fonction des coordonnées de l’axe avant d’être affichée. Pour la plupart des fonctions, on peut spécifier les propriétés du tracé par un ensemble de paires ‘paramètre’-‘valeur’, cf. la fonction `plot` de matlab.

10.1 Objets génériques

drawPoint Dessine un point ou un ensemble de points.

drawLabels Dessine des labels à des positions spécifiées. Voir aussi la fonction ‘text’ de matlab.

10.2 Objets droits

drawLine Dessine une droite.

drawEdge Dessine un segment de droite.

drawCenteredEdge Dessine un segment de droite centré sur un point, et d’angle et de longueur donnés.

drawArrow Dessine une flèche.

drawRay Dessine une demi-droite.

10.3 Polygones

drawCurve Dessine une courbe (polyligne).

drawPolygon Dessine un polygone défini par une liste de points.

fillPolygon Remplit un polygone.

drawRect Dessine un rectangle.

drawRect2 Dessine un rectangle centré.

10.4 Objets courbes

drawCircle Dessine un cercle.

drawCircleArc Dessine un arc de cercle.

drawEllipse Dessine une ellipse.

drawEllipseArc Dessine un arc d'ellipse.

drawParabola Dessine un arc de parabole.

Index

angle2points, 7
angle3Points, 7
angleSort, 7

bisector, 11

cart2geod, 17
cartesianLine, 10
centroid, 6
circleArcAsCurve, 23
circleAsPolygon, 23
clipEdge, 13
clipLineRect, 13
clipPolygon, 20
convexification, 19
crackPattern, 28
crackPattern2, 29
createCircle, 22
createDirectedCircle, 22
createEdge, 11
createLine, 10
curvature, 17
curveCentroid, 17
curveLength, 17

distancePointEdge, 14
distancePointLine, 14
distancePoints, 7
drawArrow, 33
drawCenteredEdge, 33
drawCircle, 34
drawCircleArc, 34
drawCurve, 34
drawEdge, 33
drawEllipse, 34
drawEllipseArc, 34
drawLabels, 33
drawLine, 33
drawParabola, 34
drawPoint, 33
drawPolygon, 34
drawRay, 33
drawRect, 34
drawRect2, 34

edgeAngle, 12
edgeLength, 12
ellipseAsPolygon, 23
enclosingCircle, 23

fillPolygon, 34

geod2cart, 17

hexagonalGrid, 28
homothecy, 31

inCircle, 24
intersectEdges, 13
intersectLineEdge, 13
intersectLinePolygon, 21
intersectLines, 12
invertLine, 11

- isLeftOriented, 15
- lineAngle, 12
- lineFit, 11
- linePosition, 14
- lineSymmetry, 31
- medialAxisConvex, 21
- medianLine, 10
- minDistancePoints, 8
- normalize, 8
- onCircle, 24
- onEdge, 15
- onLine, 15
- onRay, 15
- orthogonalLine, 10
- parallelLine, 10
- parametrize, 17
- pointOnLine, 14
- polarPoint, 6
- polyfit2, 26
- polygonArea, 18
- polygonCentroid, 18
- polygonClipHP, 20
- polygonExpand, 21
- polygonLength, 18
- polygonNormalAngle, 18
- polynomialCurveCentroid, 26
- polynomialCurveCurvature, 27
- polynomialCurveCurvatures, 27
- polynomialCurveDerivative, 26
- polynomialCurveFit, 25
- polynomialCurveLength, 26
- polynomialCurveNormal, 26
- polynomialCurvePoint, 26
- polynomialCurvePosition, 26
- polynomialDerivate, 27
- projPointOnLine, 14
- readPolygon, 19
- rectAsPolygon, 20
- rotation, 31
- scaling, 31
- squareGrid, 28
- steinerPoint, 19
- steinerPolygon, 20
- supportFunction, 19
- transformEdge, 32
- transformLine, 32
- transformPoint, 31
- transformVector, 31
- translation, 30
- triangleGrid, 28
- vecnorm, 8