Boîte à outil matlab pour le calcul géométrique

D. Legland

13 octobre 2008

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 je détaille le rôle et les arguments des différentes fonctions.

Table des matières

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 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.

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 un seuil de tolérance. Deux points sont donc égaux si leur distance est inférieure ou égale au seuil de tolérance. En pratique, une valeur de seuil de 10^{-14} est utilisée.

Une amélioration appréciable des fonctions consisterait à pouvoir spécifier le seuil de 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.

formatAngle
Formate un angle pour s'assurer que le résultat est compris entre 0 et 2π.

entrée la valeur de l'angle en radians

sortie l'angle, avec une valeur comprise 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 découper les primitives dans une zone donnée.

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

3 Points et vecteurs

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 pas de différence formelle entre un point et un vecteur.

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

3.1 Création de points

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.

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.

3.2 Calculs sur les points

angle2points
Calcule l'angle par rapport à l'horizontale 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.

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.

entrée les coordonnées de deux points.

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

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 les coordonnées de trois points A, B et C.

sortie l'angle formé par les vecteurs $(\overrightarrow{BA}, \overrightarrow{BC})$.

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.

distancePoints
Calcule la distance entre deux points, entre un points et un groupe de points, ou entre deux groupes de points. La métrique par défaut est la métrique euclidienne, mais on peut en spécifier une autre. Le calcul effectué est le suivant :

$$d_{AB} = \left(|x_A - x_B|^d + |y_A - y_B|^d \right)^{1/d}$$

où d est l'indicateur de métrique : 2 pour la métrique euclidienne, 1 pour la métrique Manhattan. Si $d = \infty$, la fonction renvoie la valeur max des différences absolues.

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 les mêmes indices dans les deux groupes.

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.

3.3 Manipulation de vecteurs

vecnorm
Calcule la norme d'un vecteur, donnée par $\sqrt{x^2 + y^2}$.

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

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

vectorAngle
Calcule l'angle d'un vecteur avec l'horizontale.

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

sortie l'angle du vecteur, ou un ensemble d'angles.

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

entrée un vecteur ou un ensemble de vecteurs.

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

isParallel
Teste si deux vecteurs sont parallèles.

entrée un couple de vecteurs, ou deux tableaux de vecteurs, et éventuellement un seuil de précision numérique

sortie un booléen qui vaut vrai pour chaque couple de vecteur parallèle

isPerpendicular
Teste si deux vecteurs sont perpendiculaires.

entrée un couple de vecteurs, ou deux tableaux de vecteurs, et éventuellement un seuil de précision numérique

sortie un booléen qui vaut vrai pour chaque couple de vecteur perpendiculaires

4 Droites, segments et demi-droites

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

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 : $[x_1, y_1, x_2, y_2]$.

Une généralisation possible est de considérer les « arcs de droite ». 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 $t_1 = 1$. Pour le moment cette généralisation n'est pas employée dans la bibliothèque.

4.1 Fonctions de création

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

createLine
Crée une droite à partir de deux points. L'origine de la droite est le premier point, le vecteur directeur est le vecteur joignant les deux points.

entrée les coordonnées de deux points.

sortie une représentation de la droite passant par les deux points.

parallelLine
Crée une droite parallèle à une droite donnée.

entrée la représentation 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.

orthogonalLine
Crée une droite perpendiculaire à une droite donnée.

entrée la représentation 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.

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

entrée les coordonnées de deux points.

sortie une représentation de la médiatrice des deux points.

cartesianLine
Crée une droite à partir de son équation cartésienne : $ax + by + c = 0$.

entrée les trois paramètres a, b et c.

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

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 représentation des deux droites.

sortie la bissectrice des points ou des droites.

invertLine
Inverse l'orientation d'une droite. Le changement d'orientation change la représentation 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.

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 droite, d'autre part de rendre le code plus explicite.

entrée les coordonnées de deux points.

sortie la représentation du segment.

createRay
Crée une demi-droite à partir d'un point et d'un angle.

entrée les coordonnées d'un point, et un angle.

sortie la représentation de la demi-droite ayant le point pour origine, et ayant l'angle passé en argument avec l'horizontale.

lineFit
Ajuste une droite à un ensemble de points, au sens des moindres carrés de la distance euclidienne. La représentation est choisie telle que la somme des distances au carré entre la droite et chaque point soit minimale.

entrée un ensemble de points.

sortie une représentation de la droite ajustée.

4.2 Mesures sur les droites

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.

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.

edgeLength
Calcule la longueur d'un segment de droite.

entrée la paramétrisation du segment.

sortie la longueur du segment.

4.3 Opérations géométriques

Fonctions de calculs d'intersection, et de clipping des droites et segments de droite.

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 $[\mathrm{NaN}, \mathrm{NaN}]$. Si les droites sont confondues, renvoie $[\infty, \infty]$.

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 $[\mathrm{NaN}, \mathrm{NaN}]$. Si les segments sont colinéaires, renvoie $[\infty, \infty]$.

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 $[\mathrm{NaN}, \mathrm{NaN}]$. Si les objets sont colinéaires, renvoie $[\infty, \infty]$.

clipLine
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.

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.

4.4 Interactions entre droites et points

Calculs sur les positions respectives d'un point et d'une droite ou d'un segment.

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.

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.

linePosition
Calcule la position t_p d'un point sur une droite, telle que les coordonnées du point soient égales 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).

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.

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.

4.5 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 ».

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, 0 sinon.

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, 0 sinon.

onRay
Teste si un point appartient à 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, 0 sinon.

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.

5 Polygones et polylignes

On distingue le polygone, qui est une figure fermée, qui ne se croise pas, et le chemin polygonal (ou « polyligne », de l'anglais polyline), qui est un ensemble de segments de droites connectés les uns aux autres, et qui peut éventuellement se croiser.

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 $(\mathrm{NaN}, \mathrm{NaN})$. Cependant, la plupart des fonctions ne considèrent que les polygones simples.

5.1 Fonctions sur les chemins polygonaux (polylines)

Il s'agit ici de courbes représentées par un ensemble de points. Pour des mesures plus précises, on peut consulter la section 7, qui considère des courbes polynomiales à la place d'approximations polygonales.

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

entrée les coordonnées des sommets du chemin polygonal.

sortie la longueur du chemin polygonal

curveCentroid
Calcule le centroïde de la courbe. La méthode consiste à calculer le centroïde des milieux des segments, pondérés par la longueur de chaque segment. Pour le centroïde du polygone équivalent, voir la fonction `polygonCentroid` (p. 19).

entrée les coordonnées des sommets du chemin polygonal.

sortie le centroïde du chemin polygonal.

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

entrée les coordonnées des sommets du chemin polygonal.

sortie un vecteur colonne contenant la paramétrisation correspondant à chaque sommet.

subCurve
Extrait une portion de la courbe ou du polygone. La portion est définie par les indices des points extrêmes, et par un sens de parcours (direct : sens croissant des indices, ou indirect : sens décroissant des indices).

entrée les coordonnées des sommets du chemin polygonal, l'indice du premier point, l'indice du dernier point, et éventuellement le sens de parcours.

sortie le chemin polygonal extrait de l'entrée. Si l'indice du début est supérieur à l'indice de fin, on boucle pour repartir au début.

cart2geod
Convertit des coordonnées cartésiennes en coordonnées « géodésiques », composées de la position géodésique sur la courbe, et la distance signée à la courbe.

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 pouvoir créer rapidement une courbe parallèle à une autre.

5.2 Mesures de base sur les polygones

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).

polygonArea
Calcule l'aire signée d'un 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.

polygonLength
Calcule le périmètre d'un polygone.

entrée les coordonnées des sommets du polygone.

sortie le périmètre du polygone.

polygonContains
Teste si un point est contenu dans un polygone, qui peut être troué. Il existe la fonction Matlab `inpoly`, mais qui ne marche que pour les polygones simplement connexes.

entrée les coordonnées des sommets du polygone, les coordonnées d'un point.

sortie un booléen qui vaut vrai si le point est dans le polygone.

polygonNormalAngle
Calcule l'angle normal en un ou plusieurs sommet(s) d'un polygone. Si on calcule les angles normaux de tous les sommets d'un polygone, leur somme vaut 2π.

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.

polygonBounds
Calcule la boîte englobante d'un polygone.

entrée les coordonnées des sommets d'un polygone.

sortie la plus petite boîte contenant les sommets du polygone.

5.3 Création et composition de polygones

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.

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), soit un vecteur ligne de la forme : $[x_0, y_0, w, h, \theta]$.

sortie les coordonnées des coins du rectangle.

clipPolygon
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.

clipPolygonHP
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. Il faudrait que je vérifie que l'algorithme marche bien pour des polygones non convexes...

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

sortie les coordonnées des sommets du polygone découpé.

splitPolygons
Convertit un polygone multi-connecté en une liste de polygones simplement connectés.

entrée les coordonnées des sommets d'un polygone, avec des couples [NaN ;NaN] pour marquer les limites de polygones.

sortie une liste de cellules, contenant un polygone connexe par cellule.

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.

5.4 Opérations sur les polygones

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

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

sortie les coordonnées des points d'intersection.

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.

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.

5.5 Mesures plus évoluées sur les polygones

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 géométriques (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.

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.

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.

6 Cercles, ellipses, et autres coniques

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 d 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). La représentation d'un cercle est donc la suivante :

$$[x_c, y_c, r, d]$$

Pour un arc de cercle, on oublie le paramètre d'orientation, on ajoute deux éléments correspondants à l'angle de début, et à l'angle de fin :

$$[x_c, y_c, r, \theta_1, \theta_2]$$

6.1 Fonctions de création

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.

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.

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.

inertiaEllipse
Calcule l'ellipse d'inertie d'un ensemble de points.

entrée un ensemble de points.

sortie l'ellipse d'inertie de l'ensemble de points donné en entrée.

6.2 Fonctions de conversion

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.

ellipseAsPolygon
Convertit une ellipse en polygone, éventuellement en spécifiant le nombre de sommets. La représentation d'une ellipse est la même que pour un cercle, en remplaçant le paramètre de rayon par deux paramètres a et b correspondants aux longueurs des demi axes :

$$[x_c, y_c, a, b, \theta]$$

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

sortie un polygone approchant l'ellipse.

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

entrée la représentation 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.

6.3 Fonctions de tests

onCircle
Teste si un point est situé sur un cercle.

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

7 Courbes polynomiales

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.

7.1 Fonctions de base

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

polyfit2
Fonction simplifiée d'ajustement de courbe polynomiale à un ensemble de points.

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

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

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

polynomialCurveCentroid
Calcule le centroïde d'une courbe polynomiale.

7.2 Calculs de courbures

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

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

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

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

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

8 Autres formes géométriques

Je présente ici d'autres formes géométriques : les grilles de points, et les mosaïques.

8.1 Grilles de points

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

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

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

8.2 Mosaïques

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

crackPattern2
Une variante du modèle de fissure : chaque point initie 3 fissures à 120 degrés les une des autres.

9 Transformations affines

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 pour lesquelles la dernière ligne n'est pas forcément égale à $[0, 0, 1]$ (les parallèles ne sont plus conservées).

9.1 Fonctions de création

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.

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.

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.

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

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

9.2 Transformation de formes

transformPoint
Retourne les coordonnées du ou des points passés en paramètres.

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

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

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

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. Seuls les points visibles dans la fenêtre sont dessinés.

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

10.2 Objets droits

drawLine Dessine une droite. La droite est automatiquement découpée pour tenir dans la fenêtre courante.

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 (expérimental...)

drawRay Dessine une demi-droite. La demi-droite est découpée aux dimensions de la fenêtre courante.

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