Programming project 2: Planar and solid angles

Background

This product deals with geometric concepts in $\mathbb{R}^n$, in particular in $\mathbb{R}^3$. For vectors $x, y \in \mathbb{R}^n$, let us recall of the definition of the Euclidean norm and of the scalar product (or dot product),

$$\|x\| = \Big(\sum_{0 \le j < n} x_j^2 \Big)^{1/2}, \quad x \cdot y = \sum_{0 \le j < n} x_j y_j. \tag{1}$$

For $x, y \in \mathbb{R}^3$, we can also define the vector product (or cross product)

$$x \times y = \begin{pmatrix} x_1 y_2 - x_2 y_1 \\ x_2 y_0 - x_0 y_2 \\ x_0 y_1 - x_1 y_0 \end{pmatrix} \in \mathbb{R}^3. \tag{2}$$

Note that the vector product is defined for vectors in three dimensions only, not generally in $\mathbb{R}^n$.

Given these basic notions, we can compute a number of relevant geometric quantitities. The *dihedral angle* between two vectors $x, y \in \mathbb{R}^n$ is the number $\varphi_{xy} \in [0, \pi]$ such that

$$\cos \varphi_{xy} = \frac{x \cdot y}{\|x\| \|y\|}. \tag{3}$$

The *solid angle* enclosed by three vectors $x, y, z \in \mathbb{R}^3$ is the number $\Omega_{xyz} \in [0, 2\pi)$ such that

$$\tan \frac{\Omega_{xyz}}{2} = \frac{|x \cdot (y \times z)|}{\|x\|\|y\|\|z\| + (x \cdot y)\|z\| + (x \cdot z)\|y\| + (y \cdot z)\|x\|}. \tag{4}$$

Further, two (not collinear) vectors in $\mathbb{R}^3$ define a plane through the origin, and the dihedral angle $\theta_{xy,zw} \in [0, \frac{\pi}{2}]$ between two such planes, spanned by vectors x, y and z, w respectively, is

$$\theta_{xy,zw} = \varphi_{x \times y, z \times w} \quad \text{or} \quad \theta_{xy,zw} = \varphi_{y \times x, z \times w}\,, \tag{5}$$

whichever of these two is smaller.

The purpose of the project is to develop a set of functions that compute these various angles and quantities derived from them.

Tasks

Throughout this project, all vectors (in parameters or return values) should be represented using the `double[]` type, and floating point numbers (e.g., for return values) should be of type `double`. Whenever a function takes array parameters as its input, make sure that these input arrays are not modified within the function.

(1) Basic functions

Implement the following functions in the class `Vectors`.

Name	Parameters	Quantity returned
`norm`	a vector x	$\|x\|$
`vectorProduct`	two vectors x, y	$x \times y$
`dihedralAngle`	two vectors x, y	φ_{xy}
`solidAngle`	three vectors x, y, z	Ω_{xyz}
`angleBetweenPlanes`	four vectors x, y, z, w	$\theta_{xy,zw}$

For all functions, take specific care of exceptional values in the input: If any of the input data is invalid, the return value should be `null` (for functions that return a vector) or `Double.NaN` (for functions that return a number). The input data is invalid if any of the input vectors is `null`, or if the vector dimensions do not match what is expected in the definitions (1)–(5).

You can ignore the case that any of the denominators in the expressions (3)–(5) might be zero, or that any number there is out of the range of the relevant trigonometric functions due to roundoff errors.

We take the convention that the norm and scalar product of vectors with no components ($n = 0$) is 0.

(2) An identity

It is claimed that for any $x, y, z \in \mathbb{R}^3$,

$$\Omega_{xyz} = \theta_{xy,yz} + \theta_{xz,yz} + \theta_{xy,xz} - \pi. \tag{6}$$

Let us verify this equation (with our conventions for the angles) for different values of x, y, z. To that end, within the class `ProposedIdentity`:

(a) Implement a function `formulaVerification`, which takes three vectors x, y, z as parameters, evaluates the expression

$$\pi + \Omega_{xyz} - \theta_{xy,yz} - \theta_{xz,yz} - \theta_{xy,xz} \tag{7}$$

and returns the resulting number as `double`. The function should return `Double.NaN` whenever the expression cannot be evaluated (cf. also part 1).

(b) Implement a function `correctnessRatio` that does the following n times (where n is an integer parameter to the function, and you can assume $n > 0$):

- It chooses three vectors $x, y, z \in \mathbb{R}^3$ at random. "Choosing at random" means that each component of the vectors is chosen independently according to a uniform distribution on the interval $[-1, 1]$. (Such a random number can be obtained with the expression `2.0*Math.random() - 1.0` .)
- It evaluates the expression (7) for those x, y, z and checks whether the result is zero up to a reasonable roundoff error. More precisely, it checks whether the absolute value of the expression is smaller than 10^{-8}.

The function then counts the number of times k that the expression is zero in this sense, and returns the ratio k/n as a floating point number, i.e., the relative frequency by which the formula (6) is correct.

(3) Finding the smallest angle

Your task here is to write a function that finds, among a list of k vectors $v^{(0)}, v^{(1)}, \ldots, v^{(k-1)} \in \mathbb{R}^3$, those three vectors which span the smallest possible solid angle. More precisely:

Within the class `MinFinder`, implement a function `minimumSolidAngle`. It takes an array `v` of type `double[][]` as input, which is interpreted as a list of vectors (e.g., `v[0]` would be of type `double[]` and be interpreted as the vector $v^{(0)}$ in the list, `v[0][2]` would be of type `double` and be interpreted as the component $v_2^{(0)}$, etc.). It finds among all combinations of three vectors in the list `v` those three vectors x, y, z (with pairwise *different* indices in the list) for which Ω_{xyz} has the smallest value.

The function should return its result in a composite data type `SolidAngleResult` which has the following fields:

Name	Type	Content
`angle`	`double`	the value of the minimum solid angle
`x`	`double[]`	the first vector of the three that span this minimum angle
`y`	`double[]`	the second vector of the three that span this minimum angle
`z`	`double[]`	the third vector of the three that span this minimum angle

The three vectors x, y, z should be ordered as they appeared in the original list `v`, i.e., x has the lowest index and z the highest among the three vectors.

In case of exceptional input values, the function should return `null`. This includes the case where any input vector or array is `null`, where an input vector is of incorrect dimension, or where the list `v` has fewer than 3 elements.

(4) Documentation

Within the source code, add Javadoc comments to every function, to every class, and to every field of composite data types. In these comments, describe the purpose of the function, class or field, and document all parameters and return values. Also, describe the handling of any special or exceptional input values, and any assumptions made on the parameter ranges.

How to prepare your submission

Start by downloading the template files from Moodle. The Java classes that you will work with are already defined there.

Read the assignment sheet carefully. Be sure to implement all functions with exactly the names, parameters and return types that are specified, and with parameters in the order as given.

While preparing your code, please follow these style and formatting guidelines. (This forms part of the assessment.)

- Place all code block delimiters – `{` and `}` – on separate lines.
- Within code blocks, indent the code by 4 spaces with respect to the surrounding code.
- Choose all names for variables, parameters, and functions to start with a lowercase character.

Make sure that your code compiles correctly. If the source code files that you submit do not compile (for whatever reason), you will receive zero marks for the affected parts of the project.

The code template also includes a unit test. This test does *not* check the output of your code – it only verifies whether you have declared your functions with the correct names and with the correct parameter/return types. You are advised to use the unit test to check your function declarations for any typos.

Test every piece of your code with meaningful test data (including any special or exceptional values). Verify that the output of your functions is plausible.

Once you are satisfied with the code, create a JAR file in BlueJ as follows:

- Select "Project → Create Jar file" in the menu.
- Leave "Main class" set to "none".
- ***Be sure to tick "Include source"*** and "Include BlueJ project files".
- Click "Continue" and save the file in a convenient location.

This JAR file is the one that you need to submit.

Hand-in, late submissions

Please submit your work by uploading it on Moodle before

Thursday, 20 January 2022 (week 2 Spring term), 13:00:00h.

Your submission should consist of the JAR file (see above) and of nothing else.

Late submissions will incur a penalty according to the University's standard rules: "Work which is up to one hour late will have five percent of marks deducted. After one hour, ten percent of the available marks will be deducted for each day (or part of day) that the work is late, up to a total of five days, including weekends and bank holidays, e.g., if work is awarded a mark of 30 out of 50, and

the work is up to one day late, the final mark is 25. After five days, the work is marked zero. Note, however, that the penalty cannot result in a mark less than zero."

Please be sure to press the "Submit assignment" button on Moodle before the deadline. You will receive an e-mail notification from Moodle when your assignment is submitted; if you have not received this, please assume that your submission is not yet complete.

If you are experiencing difficulties while attempting this assessment, you may be able to self-certify for a 4 day extension or apply for Exceptional Circumstances. Please see the latest Moodle post from the Assessment Administrators – subject: "Exceptional Circumstances and Self-Certification on Assessments" – to see which applies to you and email them at `maths-exams@york.ac.uk` if you have any further questions.

Marking

Your submission will be marked as follows. Partial solutions are acceptable, and will be awarded partial marks.

Code correctness (32 marks) Your code will be tested by an automated process (a collection of unit tests) to check whether it works correctly for various values of the input parameters. For each of these test cases that you pass, one mark will be awarded. Marks are awarded entirely on the criterion whether the output values of your code match the conditions specified on this assignment sheet.

Coding style (10 marks) Your source code will be read by the examiner. Marks will be awarded to reflect whether your code is readable, is appropriately structured, and follows the coding style conventions mentioned above. Marks may be subtracted for code that is functional but is implemented in a non-transparent or overly complicated way.

Source code documentation (5 marks) These marks are awarded for appropriate Javadoc comments on the functions in your code (see part 4).

In total, 47 marks are available. Your raw mark out of 47 will be moderated and scaled to the University scale 0–100. This scaled mark will contribute 25% towards your final mark for the module.

Academic integrity

To this project, the established Academic Integrity rules apply, as set out by the University. In particular, you must not copy program code or the text of the documentation from fellow students, or from other public or non-public sources. You must also not make your solution available to fellow students before the deadline.

You are allowed to re-use example code from the lectures or practicals (though, where you use it, you should include a comment to that effect in your source).