TUTORIAL 8 – Least Squares Function

This tutorial is to give an example on how to use the Least Squares Function to calculate a position based on RSSI readings from incoming packets. Least Squares is an algorithm used in multilateration techniques to localise a position based RSSI measurements. To use the Least Squares algorithm, you will need to use the Numerical Python (Numpy) library.

Installation

If numpy python library is not installed on your computer:

1) Ensure your VMWare session is connected to the network. Open a web browser to authenticate your password for outside access.

2) Install easy_install and other required packages first. Run `sudo apt-get install python-dev python-setuptools gfortran`

3) Install numpy. Run `sudo easy_install numpy`

Numpy Least Squares Function

The Numpy Least Squares function can be called as **numpy.linalg.lstsq.** From the online Numpy lstsq help page:

numpy.linalg.lstsq(*a*, *b*, *rcond=-1*)

Return the least-squares solution to a linear matrix equation.

Solves the equation *a x = b* by computing a vector *x* that minimizes the Euclidean 2-norm *|| b - a x ||^2*.

	a : array_like, shape (M, N)
	"Coefficient" matrix.
	b : array_like, shape (M,) or (M, K)
Parameters :	Ordinate or "dependent variable" values. If *b* is two-dimensional, the least-squares solution is calculated for each of the *K* columns of *b*.
	rcond : float, optional
	Cut-off ratio for small singular values of *a*. Singular values are set to zero if they are smaller than *rcond* times the largest singular value of *a*.
	Sums of residues; squared Euclidean 2-norm for each column in b - a*x. If the rank of *a* is < N or > M, this is an empty array. If *b* is 1-dimensional, this is a (1,) shape array. Otherwise the shape is (K,).
Returns :	Rank of matrix *a*.
	s : ndarray, shape (min(M,N),)
	Singular values of *a*.

Example

Usage example from the online Numpy lstsq [help page](#):

```
# Fit a line, ``y = mx + c``, through some noisy data-points:
import numpy as np

x = np.array([0, 1, 2, 3])
y = np.array([-1, 0.2, 0.9, 2.1])

# By examining the coefficients, we see that the line should
have a
gradient of roughly 1 and cut the y-axis at, more or less, -1.

# We can rewrite the line equation as ``y = Ap``, where ``A =
[[x 1]]``
# and ``p = [[m], [c]]``.  Now use `lstsq` to solve for `p`:

A = np.vstack([x, np.ones(len(x))]).T
A
# array([[ 0.,  1.],
# [ 1.,  1.],
# [ 2.,  1.],
# [ 3.,  1.]])

m, c = np.linalg.lstsq(A, y)[0]
print m, c
# 1.0 -0.95
```

References

1. [Numpy Least Squares Function](#)