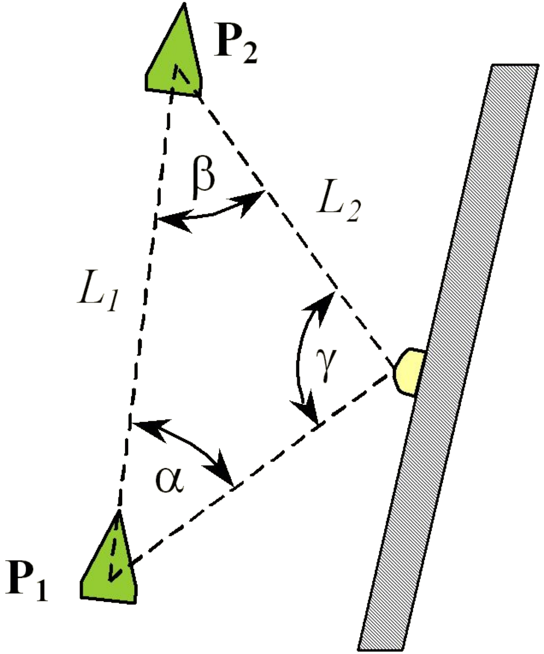
Robótica Móvel

Robot Localization - Part 2

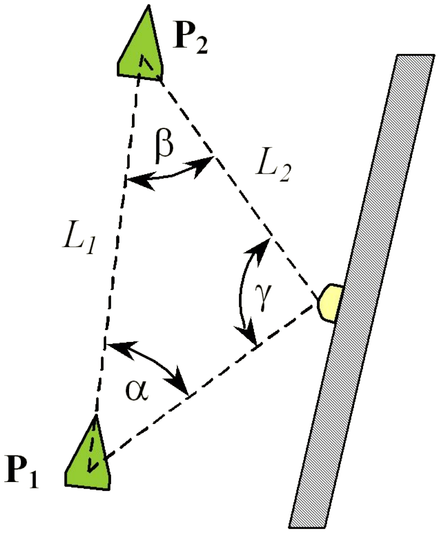
1-Extract position from motion-based triangulation

- Consider a beacon in a wall with coordinates B=(0,5) (in meters)
- Consider a differential drive robot with initial position at P1=(2,0,50º).
- Consider the robot kinematics to be the following:
  - $x(n+1)=x(n)+V*\Delta t*\cos(\theta(n))$
  - $y(n+1)=y(n)+V*\Delta t*\sin(\theta(n))$
  - $\theta(n+1)=\theta(n)+\omega*\Delta t$
- Where V=2 m/s, ω=0, Δt=0.2s, during 5 seconds.
- The goal is to simulate the robot motion by calculating the position at each iteration (after t=0.2) both by odometry and triangulation.
  - Odometry: use (x(n) ,y(n)) to plot position
  - Triangulation: use (L_2(n), β(n)) to obtain (x(n) ,y(n)) and plot it
- Follow instructions on next pages



1a-Create beacon heading function

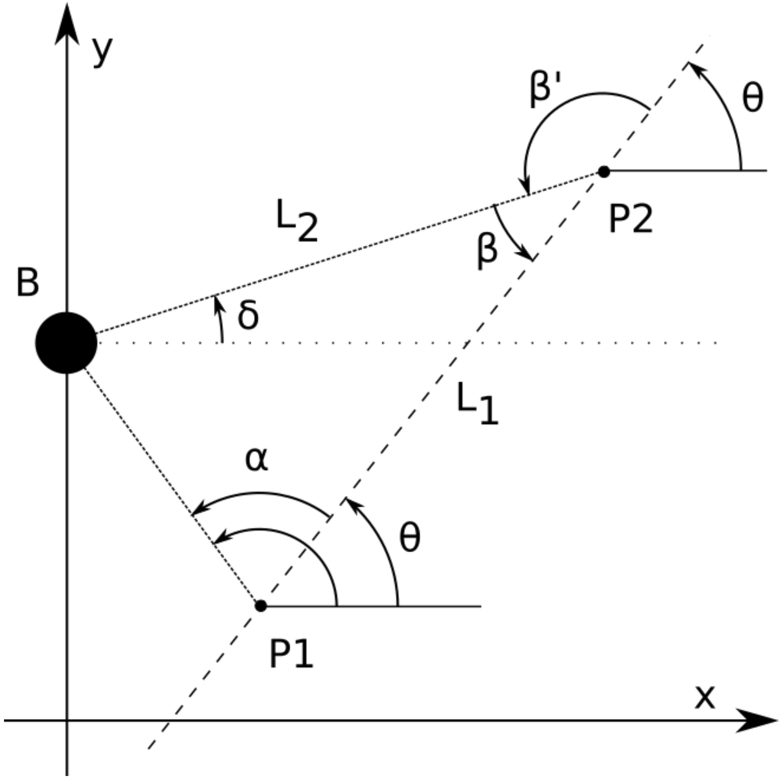
- Create the function **getbdir()** to simulate the detection of the beacon heading:
  - Accepts beacon position B, the current point P and θ.
  - Returns the angle defined by <B,P,Q> where Q is a point along the path. This angle must be < 180º !
- function a=getbdir(B,P,th)
  - B - beacon coordinates
  - P - current position in path
  - th - direction of motion in the global frame
  - a - angle <B,P,Q> mod 180º
- The function is expected to return the angle α
  - If the caller needs the β angle, an extra operation is required as explained next.



1b-Auxiliary elements to calculate angles and lines

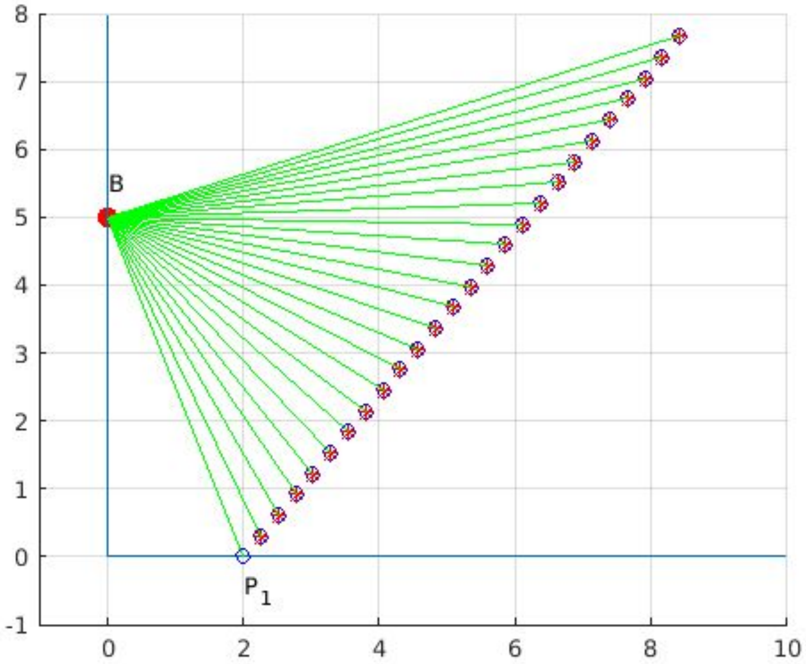
- Angle α should be obtained for initial position P1
- Angle β is obtained on subsequent calls, but what is returned is not β but β'. So, an extra operation is required :-)
- To simulate the calculation of the localization, the value of L_2 is to be used along with angle δ, so δ must be obtained after other angles.

$$L_2 = L_1 \frac{\sin \alpha}{\sin(\alpha + \beta)}$$



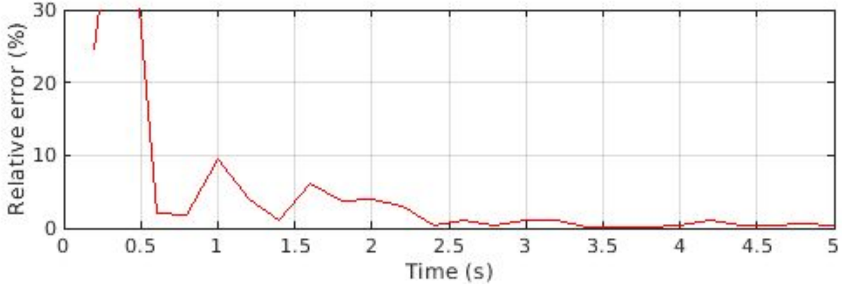
1c - Confirm the localization calculation

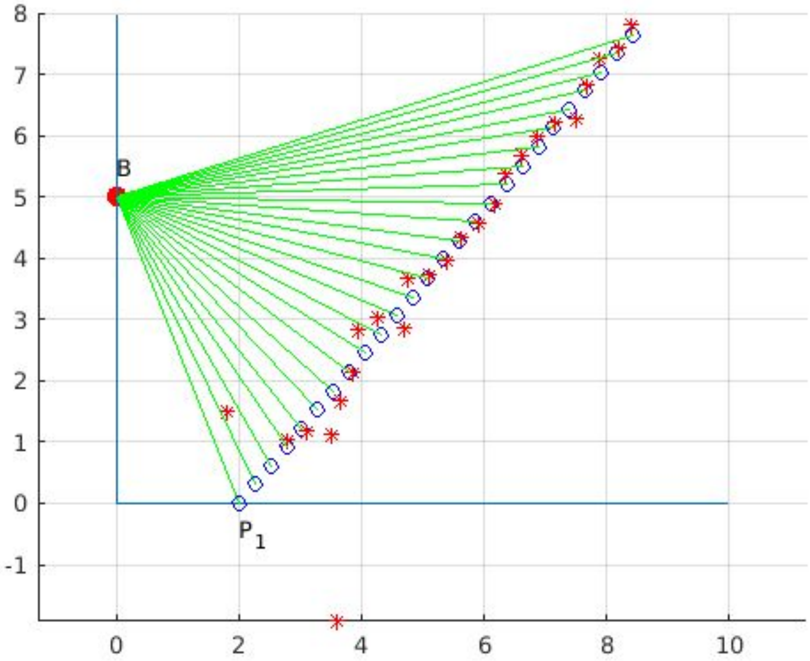
- Plot both the:
  - odometry (blue circle)
  - and the localization results (red asterisk)
- Both marks should coincide.
- There is no localization for initial position because no motion has yet occurred
- You can also draw the lines connecting each position to the beacon...



1d - Simulate uncertainty in the heading measurement

- Adjust the **getbdir()** function to introduce a gaussian error with zero mean and sigma=0.0175 rad (~1º)
- Plot the localization result in that circumstance.
- Plot also the relative errors of distances to the beacon along the path:
    - It is clear that the error decreases as path length increases





2- Localization with 2 beacons - using heading

- Using the symbolic representation in Matlab (install the Symbolic Toolbox if needed) , establish the analytical solution for the 2 beacon localization problem defined by the next equation:

$$\begin{bmatrix} -\sin\theta_1 & \cos\theta_1 \\ -\sin\theta_2 & \cos\theta_2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} -x_1\sin\theta_1 + y_1\cos\theta_1 \\ -x_2\sin\theta_2 + y_2\cos\theta_2 \end{bmatrix}$$

- Hints: create the symbolic variables
  - syms th1 th2 x y x1 y1 x2 y2 real
  - The ‘real’ modifier after the variables limits solutions to real quantities!
- Express the solution $P=[x\ y]^T$, as the inverse of the leftmost matrix multiplied by the matrix on the right above.

2a - Confirm the results

- Apply the "**simplify**" operation and after using the "**pretty(P)**" command, verify the symbolic result obtained.
- You can also generate a LaTeX encoding of the equation with:
  - **latex(P)**
- Select the resulting LaTeX string and render it in an online service https://quicklatex.com/ and check the final result:

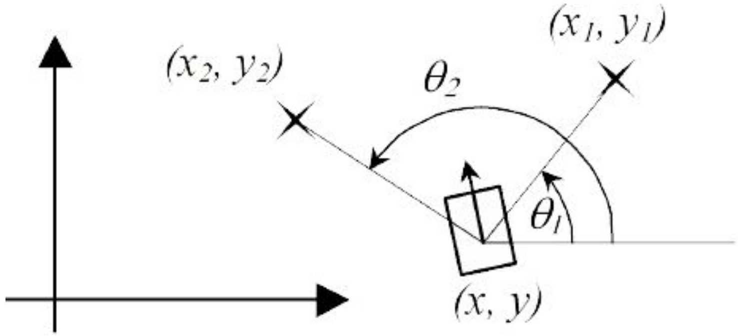
```
/   cos(th1) #1      cos(th2) #2   \
| -------------- - -------------- |
| sin(th1 - th2)   sin(th1 - th2) |
|                                 |
|   sin(th1) #1      sin(th2) #2   |
| -------------- - -------------- |
\ sin(th1 - th2)   sin(th1 - th2) /

where
   #1 == y2 cos(th2) - x2 sin(th2)
   #2 == y1 cos(th1) - x1 sin(th1)
```

$$\left(\begin{array}{c} \frac{\cos(\mathrm{th}_1)\,(y_2\,\cos(\mathrm{th}_2)-x_2\,\sin(\mathrm{th}_2))}{\sin(\mathrm{th}_1-\mathrm{th}_2)}-\frac{\cos(\mathrm{th}_2)\,(y_1\,\cos(\mathrm{th}_1)-x_1\,\sin(\mathrm{th}_1))}{\sin(\mathrm{th}_1-\mathrm{th}_2)}\\ \frac{\sin(\mathrm{th}_1)\,(y_2\,\cos(\mathrm{th}_2)-x_2\,\sin(\mathrm{th}_2))}{\sin(\mathrm{th}_1-\mathrm{th}_2)}-\frac{\sin(\mathrm{th}_2)\,(y_1\,\cos(\mathrm{th}_1)-x_1\,\sin(\mathrm{th}_1))}{\sin(\mathrm{th}_1-\mathrm{th}_2)} \end{array}\right)$$

- For more pleasant results you can replace the variable 'th' by 'theta' in the matlab program and the corresponding greek letter would be rendered in LaTeX!

2b - Confirm the solution with a concrete example

- Consider 2 beacons at the following points:
  - B1=[10; 6]; % x1,y1
  - B2=[5; 5];   % x2,y2
- And the test point
  - P=[8;2]; %x, y
- Define θ_1 and θ_2 after and B1, B2 and P (use **atan2()** function)
- Apply the symbolic solution calculated before to these values, and confirm the result as being the point P defined above!
  - See next slide for instructions on how to convert symbolic to numeric values



2c - Procedure to instantiate symbolic expressions

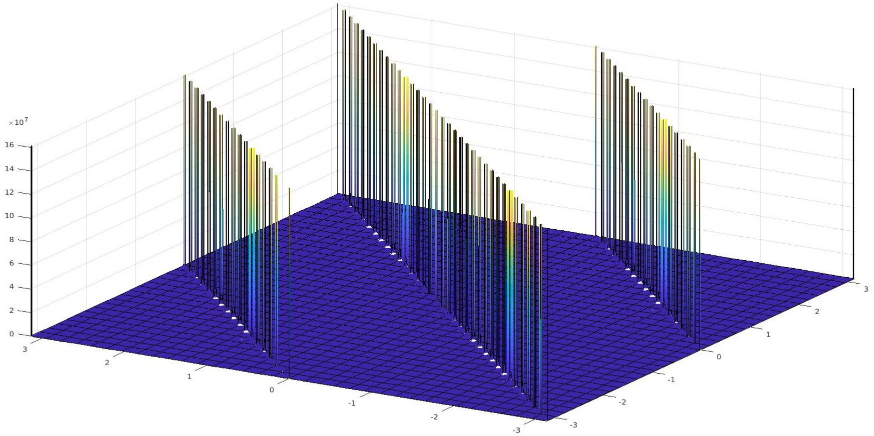
- Define the numeric values for all the variables involved.
- In this case we have:
  - `th1=... %complete with the simulate measurement for th1`
  - `th2=... %complete with the simulate measurement for th2`
  - `x1=B1(1);`
  - `y1=B1(2);`
  - `x2=B2(1);`
  - `y2=B2(2);`
- `Pf=subs(P)` % generates a numeric value, but with too much precision!
- `Pf=vpa(subs(P),3)` % forces a limited precision to 3 decimal places.

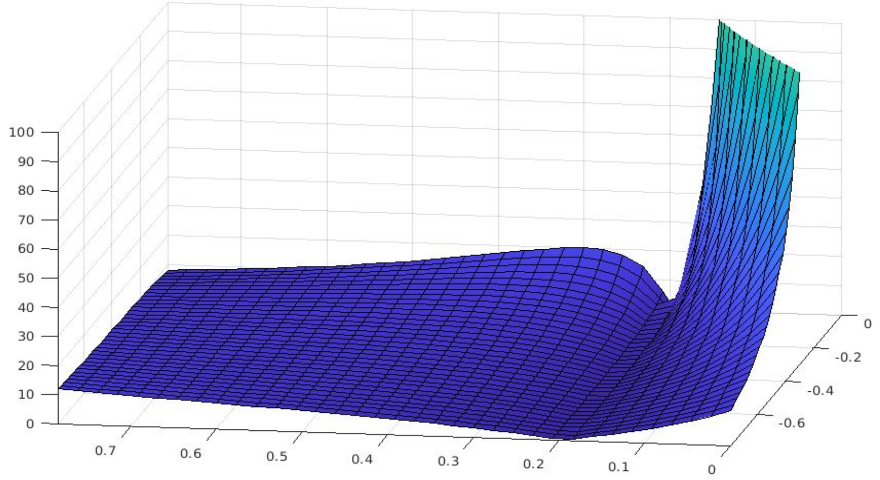
3a-Uncertainty and GDOP for the 2 beacon problem

- Using the case from the previous exercise, establish the analytical formulation of the Jacobian for the localization function.
- The functions are [x y] and the variables [th1 th2]. Use the matlab **jacobian()** function for the operation.
- Verify that for an uncertainty of $d\theta=[0.1\ 0.1]^T$ the uncertainty in the localization for the beacons and position involved is approximately $dr=[0.53\ \ 0.07]^T$
- Verify that the GDOP for this position and these beacons is approximately
    - GDOP ~ 20.0

3b - GDOP for a wide range of angles

Plot the GDOP for these intervals of angles: [-180º 180º] x [-180º 180º]   and [-45º 0] x [0 45º]





Whenever $\tan(\theta_1) = \tan(\theta_2)$, the uncertainty is unlimited!

4- Localization with n beacons using distances

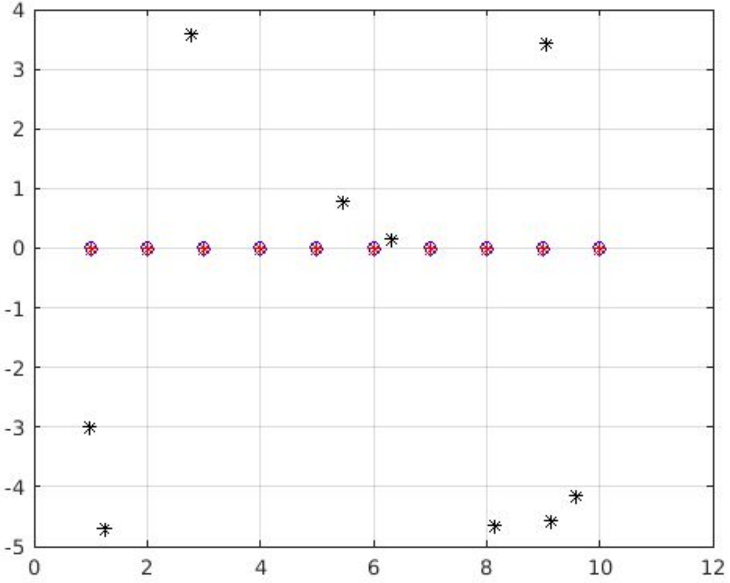
- Consider a setup with n beacons spread randomly in the plane within this region: [0 10] in x, and [-5 5] in y.
- Consider a robot moving from [0 0] to [10 0]
- Plot the calculated localization of the robot along the path

$$\mathbf{X} = \mathbf{A}^{+} \cdot \mathbf{B} = \left(\mathbf{A}^{\intercal} \cdot \mathbf{A}\right)^{-1} \mathbf{A}^{\intercal} \cdot \mathbf{B}$$

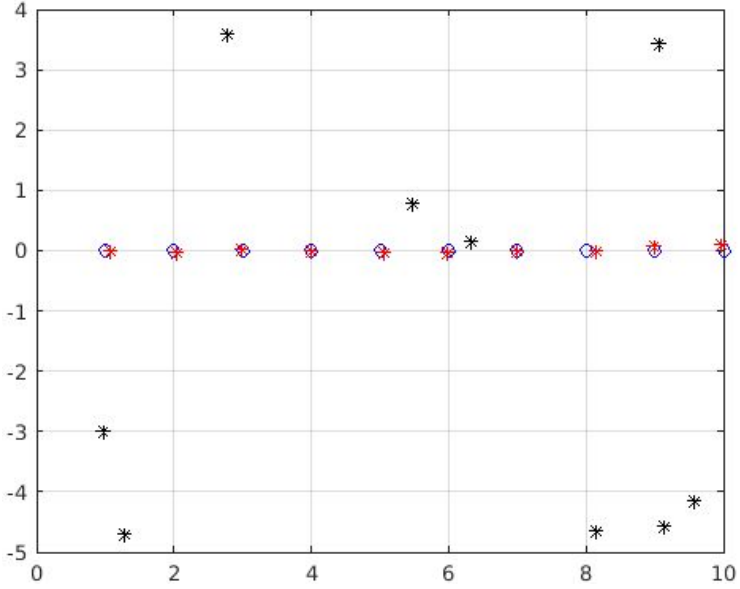
$$\mathbf{A} = \begin{bmatrix} 2\left(x_1 - x_2\right) & 2\left(y_1 - y_2\right) \\ 2\left(x_1 - x_3\right) & 2\left(y_1 - y_3\right) \\ \vdots & \vdots \\ 2\left(x_1 - x_n\right) & 2\left(y_1 - y_n\right) \end{bmatrix} \qquad \mathbf{B} = \begin{bmatrix} d_2^2 - d_1^2 + x_1^2 - x_2^2 + y_1^2 - y_2^2 \\ d_3^2 - d_1^2 + x_1^2 - x_3^2 + y_1^2 - y_3^2 \\ \vdots \\ d_n^2 - d_1^2 + x_1^2 - x_n^2 + y_1^2 - y_n^2 \end{bmatrix}$$

4b-Illustration of cases with n=9, with and without noise

No noise in measurements



Gaussian noise with sigma=0.1



rng(0) in both cases