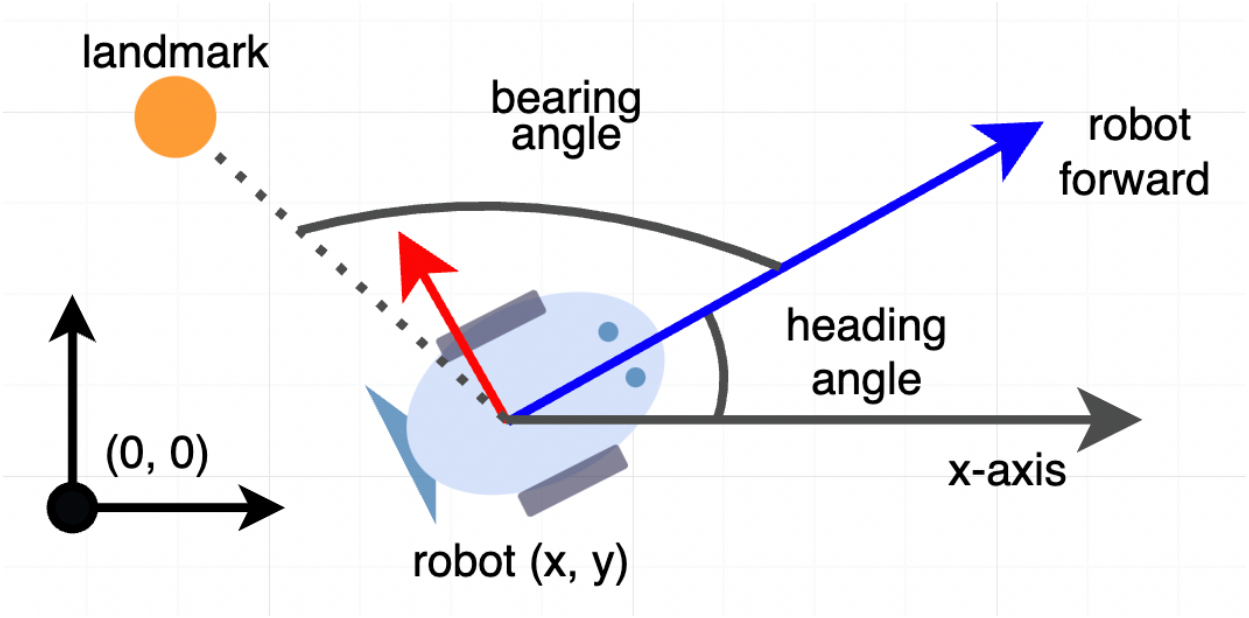
Symforce Tutorial

Notebook ini adalah tutorial untuk menggunakan Symforce. Tutorial ini akan membahas tentang:

- Instalasi Symforce menggunakan pip via bash command
- Contoh sederhana pemodelan dan penyelesaian masalah pengoptimalan dengan SymForce. Dalam contoh ini robot bergerak melalui bidang 2D dan tujuannya adalah untuk memperkirakan posenya pada beberapa langkah waktu dengan pengukuran kebisingan.



Instalasi Symforce menggunakan pip via bash command

Note : tested menggunakan Github Codespace dengan Ubuntu 20.04

In [26]:
```
%%bash
pip install symforce
```

```
Requirement already satisfied: symforce in /usr/local/python/3.10.4/lib/
python3.10/site-packages (0.7.0)
Requirement already satisfied: skymarshal==0.7.0 in /usr/local/python/3.10.4/lib/
python3.10/site-packages (from symforce) (0.7.0)
Requirement already satisfied: jinja2 in /home/codespace/.local/lib/python3.10/
site-packages (from symforce) (3.1.2)
Requirement already satisfied: numpy in /home/codespace/.local/lib/python3.10/
site-packages (from symforce) (1.23.4)
Requirement already satisfied: black in /usr/local/python/3.10.4/lib/python3.10/
site-packages (from symforce) (22.10.0)
Requirement already satisfied: graphviz in /usr/local/python/3.10.4/lib/
python3.10/site-packages (from symforce) (0.20.1)
Requirement already satisfied: clang-format in /usr/local/python/3.10.4/lib/
python3.10/site-packages (from symforce) (15.0.4)
Requirement already satisfied: sympy~=1.11.1 in /usr/local/python/3.10.4/lib/
python3.10/site-packages (from symforce) (1.11.1)
Requirement already satisfied: scipy in /home/codespace/.local/lib/python3.10/
site-packages (from symforce) (1.9.3)
Requirement already satisfied: symforce-sym==0.7.0 in /usr/local/python/3.10.4/
lib/python3.10/site-packages (from symforce) (0.7.0)
Requirement already satisfied: argh in /usr/local/python/3.10.4/lib/python3.10/
site-packages (from skymarshal==0.7.0->symforce) (0.26.2)
Requirement already satisfied: ply in /usr/local/python/3.10.4/lib/python3.10/
site-packages (from skymarshal==0.7.0->symforce) (3.11)
Requirement already satisfied: six in /home/codespace/.local/lib/python3.10/site-
packages (from skymarshal==0.7.0->symforce) (1.16.0)
Requirement already satisfied: mpmath>=0.19 in /usr/local/python/3.10.4/lib/
python3.10/site-packages (from sympy~=1.11.1->symforce) (1.2.1)
Requirement already satisfied: pathspec>=0.9.0 in /usr/local/python/3.10.4/lib/
python3.10/site-packages (from black->symforce) (0.10.2)
Requirement already satisfied: platformdirs>=2 in /home/codespace/.local/lib/
python3.10/site-packages (from black->symforce) (2.5.4)
Requirement already satisfied: mypy-extensions>=0.4.3 in /usr/local/python/
3.10.4/lib/python3.10/site-packages (from black->symforce) (0.4.3)
Requirement already satisfied: click>=8.0.0 in /usr/local/python/3.10.4/lib/
python3.10/site-packages (from black->symforce) (8.1.3)
Requirement already satisfied: tomli>=1.1.0 in /home/codespace/.local/lib/
python3.10/site-packages (from black->symforce) (2.0.1)
Requirement already satisfied: MarkupSafe>=2.0 in /home/codespace/.local/lib/
python3.10/site-packages (from jinja2->symforce) (2.1.1)
```

Import Library Symforce Symbolic API, yang berisi SymPy API yang ditambah, serta geometri dan tipe kamera

Pertama, memberi tahu SymForce untuk menggunakan epsilon bukan nol untuk mencegah singularitas. Ini tidak diperlukan saat bermain-main dengan ekspresi simbolik, tetapi penting untuk mengevaluasi beberapa hasil secara numerik.

In [1]:
```
import symforce
symforce.set_epsilon_to_symbol()
```

In [2]:
```
import symforce.symbolic as sym
import numpy as np
```

Buat pose 2D simbolis dan lokasi landmark. Menggunakan variabel simbolik memungkinkan kita menjelajahi dan membangun matematika dalam bentuk murni.

In [3]:
```
pose =sym.Pose2(
    t=sym.V2.symbolic('t'),
    R=sym.Rot2.symbolic('R')
)
landmark= sym.V2.symbolic('L')
```

Ubah landmark menjadi bingkai lokal robot. Representasi pose sebagai world_T_body, artinya mengambil landmark di world frame dan mendapatkan posisinya di body frame

In [23]:
```
landmark_body=pose.inverse() * landmark
```

sf.Rot2 direpresentasikan secara internal oleh bilangan kompleks (Rre, Rim) dan kita dapat mempelajari bagaimana ia merotasi landmark L.

Untuk tujuan eksplorasi, mari ambil jacobian tengara kerangka tubuh sehubungan dengan ruang singgung Pose2, dengan parameter sebagai (, x, y):

In [5]:
```
landmark_body.jacobian(pose)
```

```
[-L0*R_im + L1*R_re + t0*R_im - t1*R_re, -R_re, -R_im]
[-L0*R_re - L1*R_im + t0*R_re + t1*R_im, R_im, -R_re]
```

Perhatikan bahwa meskipun orientasi disimpan sebagai bilangan kompleks, ruang singgung adalah sudut skalar dan SymForce memahami hal itu.

Sekarang hitung sudut bantalan relatif:

In [6]:
```
sym.atan2(landmark_body[0], landmark_body[1])
```

Out[6]:
```
atan2(L0*R_re + L1*R_im - (t0*R_re + t1*R_im), -L0*R_im + L1*R_re + (0.5
+ sign(-L0*R_im + L1*R_re - (-t0*R_im + t1*R_re)))*epsilon - (-t0*R_im +
t1*R_re))
```

Satu catatan penting adalah bahwa atan2 singular pada (0, 0). SymForce menangani ini dengan menempatkan simbol ϵ (epsilon) yang mempertahankan nilai ekspresi dalam batas ϵ → 0, tetapi memungkinkan evaluasi saat runtime dengan sangat kecil nilai bukan nol. Fungsi dengan singularitas menerima argumen epsilon:

In [7]:
```
sym.V3.symbolic('x').norm(epsilon=sym.epsilon())
```

Out[7]:
```
sqrt(epsilon + x0**2 + x1**2 + x2**2)
```

In [10]:
```
import symforce
try :
    symforce.set_epsilon_to_symbol()
except:
    pass
```

Keterangan

In [11]:
```
from symforce.values import Values
```

In [12]:
```
num_poses=3
num_landmarks=3
```

In [13…
```
initial_values=Values(
    poses=[sym.Pose2.identity()] * num_poses,
    landmarks=[sym.V2(-2, 2), sym.V2(1, -3), sym.V2(5, 2)],
    distances=[1.7, 1.4],
    angles=np.deg2rad([[145, 335, 55], [185, 310, 70], [215, 310, 70]]).to
    epsilon=sym.numeric_epsilon,
)
```

In [14…
```
def bearing_residual(
    pose: sym.Pose2, landmark: sym.V2, angle: sym.Scalar, epsilon: sym.Sca
) -> sym.V1:
    t_body = pose.inverse() * landmark
    predicted_angle = sym.atan2(t_body[1], t_body[0], epsilon=epsilon)
    return sym.V1(sym.wrap_angle(predicted_angle - angle))
```

In [15…
```
def odometry_residual(
    pose_a: sym.Pose2, pose_b: sym.Pose2, dist: sym.Scalar, epsilon: sym.S
) -> sym.V1:
    return sym.V1((pose_b.t - pose_a.t).norm(epsilon=epsilon) - dist)
```

In [1…
```
from symforce.opt.factor import Factor

factors = []

# Bearing factors
for i in range(num_poses):
    for j in range(num_landmarks):
        factors.append(Factor(
            residual=bearing_residual,
            keys=[f"poses[{i}]", f"landmarks[{j}]", f"angles[{i}][{j}]", "e
        ))

# Odometry factors
for i in range(num_poses - 1):
    factors.append(Factor(
        residual=odometry_residual,
        keys=[f"poses[{i}]", f"poses[{i + 1}]", f"distances[{i}]", "epsilor
    ))
import warnings
warnings.filterwarnings("ignore")
```

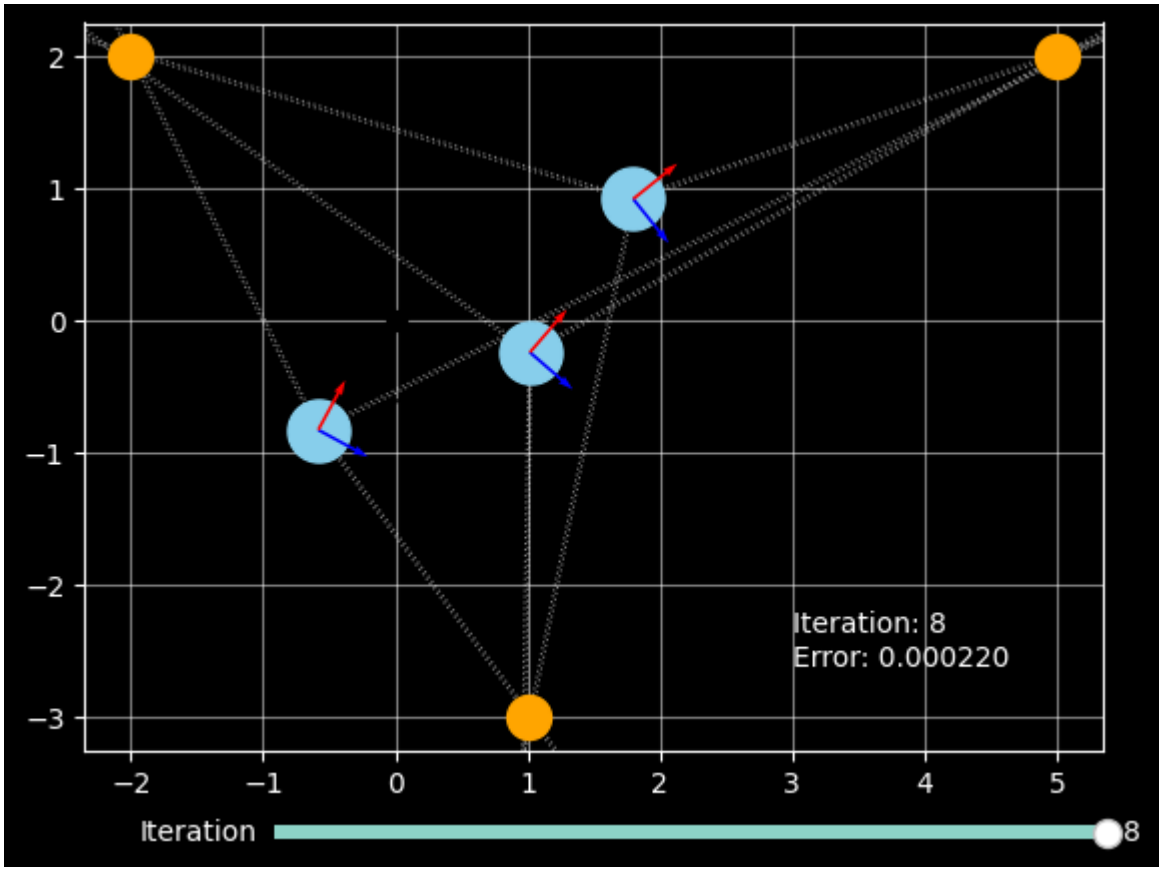
In [17…
```
from symforce.opt.optimizer import Optimizer

optimizer = Optimizer(
    factors=factors,
    optimized_keys=[f"poses[{i}]" for i in range(num_poses)],
    # So that we save more information about each iteration, to visualize
    debug_stats=True,
)
```

In [18]:
```
result = optimizer.optimize(initial_values)
```

```
[2022-12-03 10:13:12.447] [info] LM<sym::Optimize> [iter    0] lambda:
1.000e+00, error prev/linear/new: 5.143/2.872/2.203, rel reduction: 0.57166
[2022-12-03 10:13:12.448] [info] LM<sym::Optimize> [iter    1] lambda:
2.500e-01, error prev/linear/new: 2.203/0.087/0.074, rel reduction: 0.96655
[2022-12-03 10:13:12.449] [info] LM<sym::Optimize> [iter    2] lambda:
6.250e-02, error prev/linear/new: 0.074/0.006/0.006, rel reduction: 0.91401
[2022-12-03 10:13:12.449] [info] LM<sym::Optimize> [iter    3] lambda:
1.562e-02, error prev/linear/new: 0.006/0.001/0.001, rel reduction: 0.90323
[2022-12-03 10:13:12.450] [info] LM<sym::Optimize> [iter    4] lambda:
3.906e-03, error prev/linear/new: 0.001/0.000/0.000, rel reduction: 0.60930
[2022-12-03 10:13:12.450] [info] LM<sym::Optimize> [iter    5] lambda:
9.766e-04, error prev/linear/new: 0.000/0.000/0.000, rel reduction: 0.08144
[2022-12-03 10:13:12.451] [info] LM<sym::Optimize> [iter    6] lambda:
2.441e-04, error prev/linear/new: 0.000/0.000/0.000, rel reduction: 0.00012
[2022-12-03 10:13:12.453] [info] LM<sym::Optimize> [iter    7] lambda:
6.104e-05, error prev/linear/new: 0.000/0.000/0.000, rel reduction: 0.00000
```

In [19…

```
from symforce.examples.robot_2d_localization.plotting import plot_solutior
plot_solution(optimizer, result)
```



Generate C++ Code

Untuk setiap faktor, SymForce mengintrospeksi bentuk fungsi simbolik, melewati input simbolik untuk membuat ekspresi output, secara otomatis menghitung jacobian dari ekspresi output tersebut dengan variabel yang dioptimalkan, dan menghasilkan kode runtime cepat untuk variabel tersebut. Class Codegen adalah alat utama untuk menghasilkan kode runtime dari ekspresi simbolik. Dalam hal ini, kami meneruskan fungsi residual_bearing dan mengonfigurasinya untuk menghasilkan kode C++:

In [20]:
```
from symforce.codegen import Codegen, CppConfig

codegen = Codegen.function(bearing_residual, config=CppConfig())
```

In [21]:
```
codegen_linearization = codegen.with_linearization(
    which_args=["pose"]
)
import warnings
warnings.filterwarnings("ignore")
```

In [22]:
```
metadata = codegen_linearization.generate_function()
# with open('coba.cpp', 'w') as f:
#     f.write(metadata.generated_files[0])
#     f.close()
# with open(metadata.generated_files[0]).read() as f:
#     lines = f.readlines()
#     lines = [l for l in lines if "ROW" in l]
#     with open("out.txt", "w") as f1:
#         f1.writelines(lines)
# print(type(metadata.generated_files[0]))
code=open(metadata.generated_files[0]).read()
with open('coba.cpp', 'w') as f:
    f.write(code)
# print(open(metadata.generated_files[0]).read())
```

In [2…
```
%%bash
wget https://raw.githubusercontent.com/symforce-org/symforce/main/gen/cpp/sy
```

```
--2022-12-03 10:13:14--  https://raw.githubusercontent.com/symforce-org/symforce/
main/gen/cpp/sym/pose2.h
Resolving raw.githubusercontent.com (raw.githubusercontent.com)...
185.199.111.133, 185.199.108.133, 185.199.109.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|
185.199.111.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 8015 (7.8K) [text/plain]
Saving to: ‘./sym/pose2.h.1’

     0K .......                                               100% 62.3M=0s

2022-12-03 10:13:14 (62.3 MB/s) - ‘./sym/pose2.h.1’ saved [8015/8015]
```