Factorization

f(x + h, y + k) variables

Decomposition

```
variable subzero = 0.0012
variable h = 2.12
variable y = 1.25
variable k = 2.00
```

f(x,y) + [hf'x(x,y) + KF'y(x,y) = 0

```
import Tfidf Vectorizer, CountVectorizer
From sklearn.decomposition import NMK, lateratDirichletAllocation
```

[Compute the qubits and max-min axis]

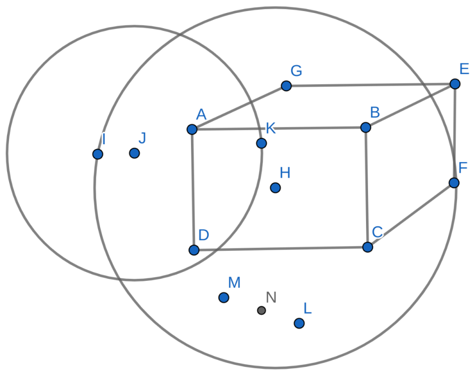
F (x,y,y',..., y(n))=0
(Pag 585)

Models

```
Matrix initialization store with the function value' => f(b) - f(a) = f'(c)
ma = np.matrix([1.0, 2.0, 3.0] three dimension states
mb = np.zeros((3,2), dtype=np.int32) #3 * 2
mc = np.array([1.0, 2.0, 3.0], [4.0, 5.0, 6.0]]) #2 * 3
md=  np.matrix([7.0, 8.0, 9.0]]) #1 * 3
Print (ma)
Print (mb)
```

```
    1) A > 0
    2) A < 0
    3) A = 0
```

*Regression log() [-40] corresponds to different dimensional states 4 * 4 float64 matrix my_det() is -40 with np.linalg.det() higher dimensions with matrices; Print mb => return #functional datasets in a Max-min margin diagram and Qubits = (a) XOR(different states) (gates b,c)*



```
Begin array demo
Creating array arr using np.array() and list with hard-coded values
Cell element type is float64
Printing array arr using built-in print()
[ 1. 3. 5. 7. 9.]
Creating int array arr using np.arange(9)
Printing array arr using built-in print()
[0 1 2 3 4 5 6 7 8]
Printing array arr using my_print() with cols=4 and dec=0
0 1 2 3
4 5 6 7 8
Creating array arr using np.zeros(5)
Printing array arr using built-in print()
[ 0. 0. 0. 0. 0.]
Creating array arr using np.linspace(2., 5., 6)
```

Grouping the datasets and give them some mechanical hierarchy models, print the elements and extract the data For aerodynamics and robotic bodies are some of the priorities of this paper.

-Functional approach is needed as well as automotive instructions to draw some practical examples that can apply to spybotics and Pydy libraries to model the robotic kinematics and find new projects behind the scenes. My aim as a programmer is apply the Mathematical models and some examples in higher order dimensions with Scikit in robots to expand the capabilities that the field of robotics in python present and give them more analysis and new characteristics in a programming level foundation with python.

```
#Robotics and (n - 1) dimensions
n_samples = 2000
n_features = 1000
n_topics = 10
n_top_elements = 20 #dimensions applied to robotic spaces
subzero = 0.0012
n= 1.12 …
#Load the data
#compare the measurements in higher dimensions
#From n variables
```

Robotic vectors (definition)

#EXAMPLES Create the Robot Model from pybotics.robot import Robot from pybotics.predefined_models import ur10

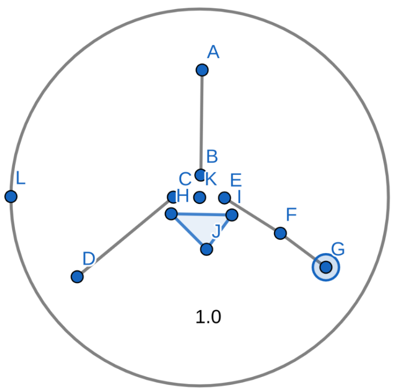
robot = Robot.from_parameters(ur10()) Define the Forces/Torques Acting on the TCP

#Calculate Joint Torques What are the joint torques required to counteract this payload? This calculation can be repeated at each discrete pose in a trajectory for trajectory dynamics The degrees of freedom of the system are n + 1, i.e. one for each pendulum link and one for the lateral motion of the cart.

M x' = F, where x = [u0, ..., un+1, q0, ..., qn+1]

The joint angles are all defined relative to the ground where the x axis defines the ground line and the y axis points up. The joint torques are applied between each adjacent link and between the cart and the lower link where a positive torque corresponds to positive angle.

*forces = [0, 0, 10] torques = [0, 0, 0] wrench = [*forces, *torques]*



instructions for robots:

-kinematics
-movements
-rotation
-calibration

```
#!/usr/bin/python3

>>> from pybotics.robot import Robot
>>> from pybotics.predefined_models import ur10
>>> robot = Robot.from_parameters(ur10())
>>> forces = [0, 0, 10]
>>> torques = [0, 0, 0]
>>> wrench = [*forces, *torques]
>>> print(robot)
Robot(kinematic_chain=MDHKinematicChain(_links=[RevoluteMDHLink(alpha=0.0,
a=0.0, theta=0.0, d=118.0), RevoluteMDHLink(alpha=1.5707963267948966, a=0.0,
theta=3.141592653589793, d=0.0), RevoluteMDHLink(alpha=0.0, a=612.7,
theta=0.0, d=0.0), RevoluteMDHLink(alpha=0.0, a=571.6, theta=0.0, d=163.9),
RevoluteMDHLink(alpha=-1.5707963267948966, a=0.0, theta=0.0, d=115.7),
RevoluteMDHLink(alpha=1.5707963267948966, a=0.0, theta=3.141592653589793,
d=92.2)]), tool=Tool(matrix=array([[1., 0., 0., 0.],
       [0., 1., 0., 0.],
       [0., 0., 1., 0.],
       [0., 0., 0., 1.]]), mass=0, cg=array([0., 0., 0.])),
world_frame=array([[1., 0., 0., 0.],
       [0., 1., 0., 0.],
       [0., 0., 1., 0.],
       [0., 0., 0., 1.]]), random_state=RandomState(MT19937) at
0x7D110BC03640, home_position=array([0., 0., 0., 0., 0., 0.]),
_joints=array([0., 0., 0., 0., 0., 0.]), _joint_limits=array([[-3.14159265,
-3.14159265, -3.14159265, -3.14159265, -3.14159265,
        -3.14159265],
       [ 3.14159265,  3.14159265,  3.14159265,  3.14159265,  3.14159265,
         3.14159265]]))
>>> print(wrench)
[0, 0, 10, 0, 0, 0]

>>> import numpy as np
>>> robot.joins = np.deg2rad([0, 0, 0, 0, 0, 0])
>>> j_torques = robot.compute_joint_torques(wrench)
>>> print(f'Robot Joints: {robot.joints}')
Robot Joints: [0. 0. 0. 0. 0. 0.]
>>> print(f'Joint Torques: {j_torques}')
Joint Torques: [11843.     0.     0.     0.     0.     0.]
>>>
>>> robot.joints = np.deg2rad([0, -90, -90, 0, -90, 0])
>>> j_torques = robot.compute_joint_torques(wrench)
>>> print(f'Robot Joints: {robot.joints}')
Robot Joints: [ 0.         -1.57079633 -1.57079633  0.         -1.57079633  0.
]
>>> print(f'Joint Torques: {j_torques}')
Joint Torques: [-1639.  7284.  1157.  1157.     0.     0.]
```

Machine learning modeling based on calculus(Scikit)

We need to model the fragments of n properties that are placed in the kinematics with a given number of elements And enumerate the topic index with the join function:

```
Def print_top_elements(model,feaure_names,n_top_words):
    For topic_idx, topic in enumerate(model.components_):
        Print (“topic:” %topic_idx)
        Print (“ “.join ([feature_names[i]
                         For i in topic.argsort()[:-n_top_words -1 : -1]))

#!/usr/bin/python3
#A Robot-model, at a very minimum, is a kinematic chain
#The kinematic chain is defined by a series of parameters
#The forward kinematics (FK) refer to the use of the kinematic equations of a
robot to compute the pose of the end-effector
(i.e., 4x4 transform matrix) from specified values for the joint parameters
(i.e., joint angles)

*******************************************************************************
************************************************************
```

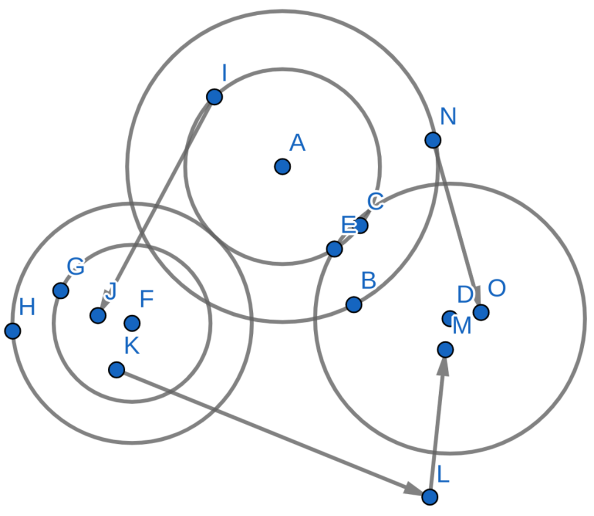
```
#The data contains outliers which are defined as observations that are far
from the others (i.e., defective TCP).
Outlier detection estimators try to fit the regions where the training data is
the most concentrated, ignoring the deviant observations.

>>> from sklearn.neighbors import LocalOutlierFactor
>>> # create classifiers and predictions
>>> localFactor = LocalOutlierFactor(contamination='auto')
>>> # predict
>>> y_pred = localfactor.fit_predict(data)

# compute accuracy to predict number of errors divided by the length of the
variable predictor multiply by one hundred possible cases
n_errors = (y_pred != truth).sum()
accuracy = (1 - n_errors / len(y_pred)) * 100
print(f"Accuracy: {accuracy:.2f}%")
```

Set of instructions for robotics:
Automate the number of elements or robot materials with a robot framework or a similar tool to print and extract functionality in order to get the total of elements that are located with some sub iteration step for automation.

1. Print_top_elements
2. N_top_elements
3. Iterate   1, 2, 3, 4, 5, 6…
4. Join elements (arg_sort()[:-n-top-words]
5. Iterate foreach_element (n_top_words)



Load the 20 newsgroups and vectorize it.
RPA implementation is a plus.
Linux (Numpy-scipy-scikit-pybotics-pydy)

Instructions:

```
print(“load the dataset…”)
t0 = time
Dataset = fetch_20newsgroups(shuffle=True, random_state =1, remove=
(‘headers’,’footers’,’quotes’))
Data_samples = dataset.data [:n_samples]
```

Functions to automate robots:

```
0.0 print function
t0  variable=time    sleep_seconds=0.00032
load_data            variable=groups       value= 1.18
mechanize            robot_speed=2.1 milliseconds
A —> B
```

```
Extract (LDA)  variable:features
tfidF = tFidF_vectorizer.fit_transform(data_samples)
print(“done in % p.3fd.”) % (time() -t0))
# Jira notes to automate robot transition to virtual environments.
tF_vectorizer = CountVectorizer(max_of=0.95,
min_dF=2,max_features=n_features,Stop_words= ‘english’)
```

```
ROBOT + MA_
Extract (LDA)
tFidF   '_vectorizer' -> transform_data()
time    '0.0032'
measure ""            -> mechanize_load_matrix()
```

#Fit the NMF model