



Enabling Semantic Search on DesignSafe: Code + Docs

Krishna Kumar

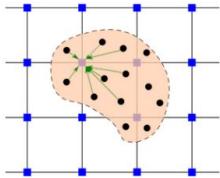
How Search on DesignSafe could help:

- Code
- Documentation

- Code search on DesignSafe is not available
- Search and find appropriate functions.
- Improve code reusability
- Search and find correct documentation
- Auto-code completion and help (Potentially inside Jupyter interface).
- Support natural language search interface.

Index search: “map nodal velocity to material points”

← → ⌂ file:///Users/krishna/research/mpm/LearnMPM/_build/html/search.html?q=map+velocity+from+nodes+to+material+points



Learn MPM

map velocity from nodes

Welcome to learn MPM with Python

An introduction to the Material Point Method

1D vibration of a single material point

Axial vibration of a 1D bar

Coding 1D MPM

Search Results

Search finished, found 5 page(s) matching the search query.

1D vibration of a single material point

An introduction to the Material Point Method

Axial vibration of a 1D bar

Update Stress First (USF)

Update Stress Last (USL)

By Krishna Kumar, Shyamini Kularathna
© Copyright 2022.

None of the results are correct!

Why index search failed?

"map velocity from nodes to material points"

Index search looks for exact matches of the words. Although this function exists in the indexed database...we use different words:

"velocity from nodes" → "nodal velocity"

"material points" → "particles"

So the search does not result in anything

This function exists!! (we just didn't find it)

```
def particle_position_velocity(mesh, dt): """ Compute particle position and velocity based on  
nodal velocity. :math:x_p += \sum_i N_i(x_p) * v_i and particle position :math:x_p += v_p * dt.
```

```
Arguments:  
    mesh: mesh  
        a mesh object  
    dt: float  
        time step  
    ....  
    for el in mesh.elements:  
        for prtcl in el.particles:  
            prtcl.velocity = 0  
            for i in range(len(el.nodes)):  
                prtcl.velocity += prtcl.shapefn[i] * el.nodes[i].velocity  
            prtcl.x += prtcl.velocity * dt
```

Index search: “velocity from nodes to particles”

The screenshot shows a search results page titled "Search Results". It displays four matching pages: "1D vibration of a single material point", "Axial vibration of a 1D bar", "An introduction to the Material Point Method", and "Update". A green box highlights the word "Update" in the sidebar, and another green box highlights the word "Search result!" in the main content area. The sidebar also lists other topics like "1D vibration of a single material point" and "Coding 1D MPM".

The screenshot shows a code editor with several search results. The first result is a function definition:

```
def nodal_total_force(mesh): """ Compute total force as sum of internal, external and damping force.
```

The second result is another function definition:

```
def nodal_acceleration_velocity(mesh, dt): """ Compute nodal acceleration as :math:a = f / m and v += a * dt
```

The third result is a function definition:

```
def particle_position_velocity(mesh, dt): """ Compute particle position and velocity based on nodal velocity. :math:x_p += \sum_i N_i(x_p) * v_i and particle position :math:x_p += v_p * dt
```

A large green box highlights the word "velocity" in the third function definition. Below the code, a bold statement reads: "The function we are looking for is at the end of the page!"

Trying with exact words in the function

It found every *match of the words*: “nodes”, “velocity” and “particles” (**not useful!**)

- Last search result and the last item in the last page

5

GPT to the rescue?

- Can we generate relevant code and documentation?
- Will it consistently generate the same output considering it is a probabilistic language model?
- Will it “lie” if it doesn’t know the solution?
- Is the solution correct?



Can ChatGPT write the code for us?

K

Write a Python code to map nodal velocities to material points in MPM 1D



The Material Point Method (MPM) is a numerical simulation technique that is used to model the behavior of materials in a wide range of applications. In MPM, the material points (MPs) are used to represent the physical behavior of the material, while the nodal velocities are used to update the positions and velocities of the MPs. Here is a basic example of how to map nodal velocities to material points in MPM 1D using Python:

Can ChatGPT write the code for us? NO!!!

```
import numpy as np

# Initialize the nodal velocities
node_velocities = np.array([1.0, 2.0, 3.0, 4.0])

# Create an array of material points
material_points = np.array([[0.2, 0.3, 0.4, 0.5], [0.1, 0.2, 0.3, 0.4]])  
← Material point location and velocities?

# Define the shape functions
def shape_function(xi):
    N = np.array([(1-xi), xi])
    return N  
← Valid only for natural coordinates

# Define the derivatives of the shape functions
def shape_function_derivative(xi):
    dNdx = np.array([-1, 1])
    return dNdx  
← Not needed!

# Iterate over all material points
for i in range(material_points.shape[1]):
    xi = material_points[0, i]
    N = shape_function(xi)
    dNdx = shape_function_derivative(xi)
    velocity = np.dot(N, node_velocities)
    material_points[1, i] += velocity  
← Incorrect as it maps every velocity from all the nodes to the material points
```

Material point location and velocities?

Valid only for natural coordinates

Not needed!

Incorrect as it maps every velocity from all the nodes to the material points

Will GPT lie if it doesn't know the answer?

K

Can you run server on demand on TACC Frontera



It is possible to run a server on demand on TACC's Frontera supercomputer, but the specifics of how to do so would depend on the type of server and the resources required. TACC provides a variety of services and tools for running and managing computing jobs on Frontera, including the Slurm job scheduler and the Lmod module system for managing software environments. Additionally, TACC provides a number of tutorials and documentation on their website to help users get started with running jobs on Frontera. It would be best to check TACC website and documentation for more information on the specific steps required to run a server on demand on Frontera.



Can we stop it from lying?

K

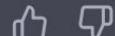
Answer the question as truthfully as possible, and if you're unsure of the answer, say "Sorry, I don't know".

Q: Can you run server on demand on TACC Frontera

A:



Sorry, I don't know.



This works! But it is not really useful, since it doesn't help us with a solution...

How can we stop GPT from hallucinating?

- Hallucination “lie” is when GPT tries to answer confidently when it doesn’t know the answer.
- Solutions it provides are sometimes inaccurate/incorrect
- ***What if we can use existing documentations and source codes hosted on DesignSafe to generate answers?***
 - ***Identify the “correct” code from a “high-quality” published code***
 - ***Cite sources for the answers***

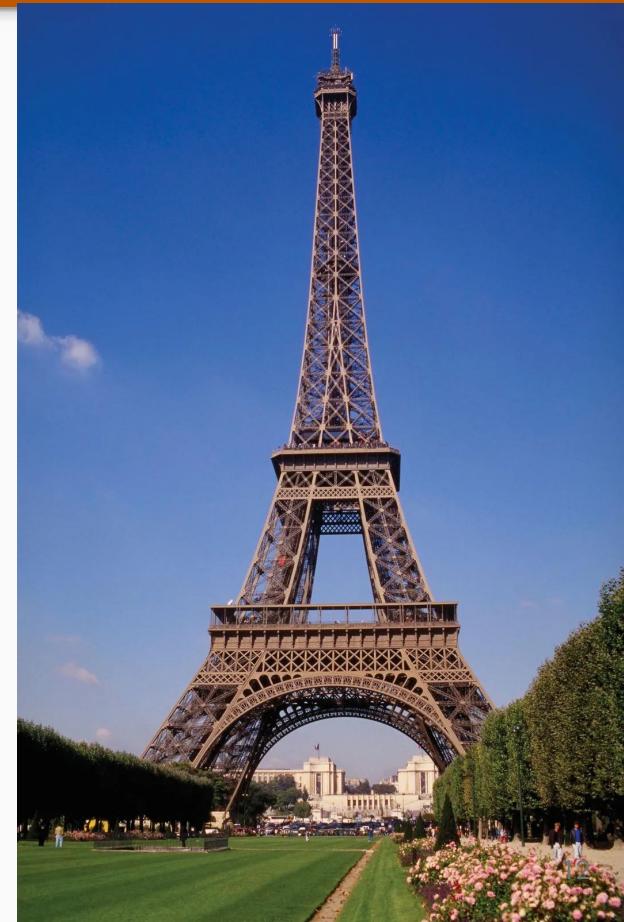
Semantic search to the rescue?

Semantic search denotes search with meaning, as distinguished from lexical search where the search engine looks for literal matches of the query words or variants of them, without understanding the overall meaning of the query

Consider searching for the following text:

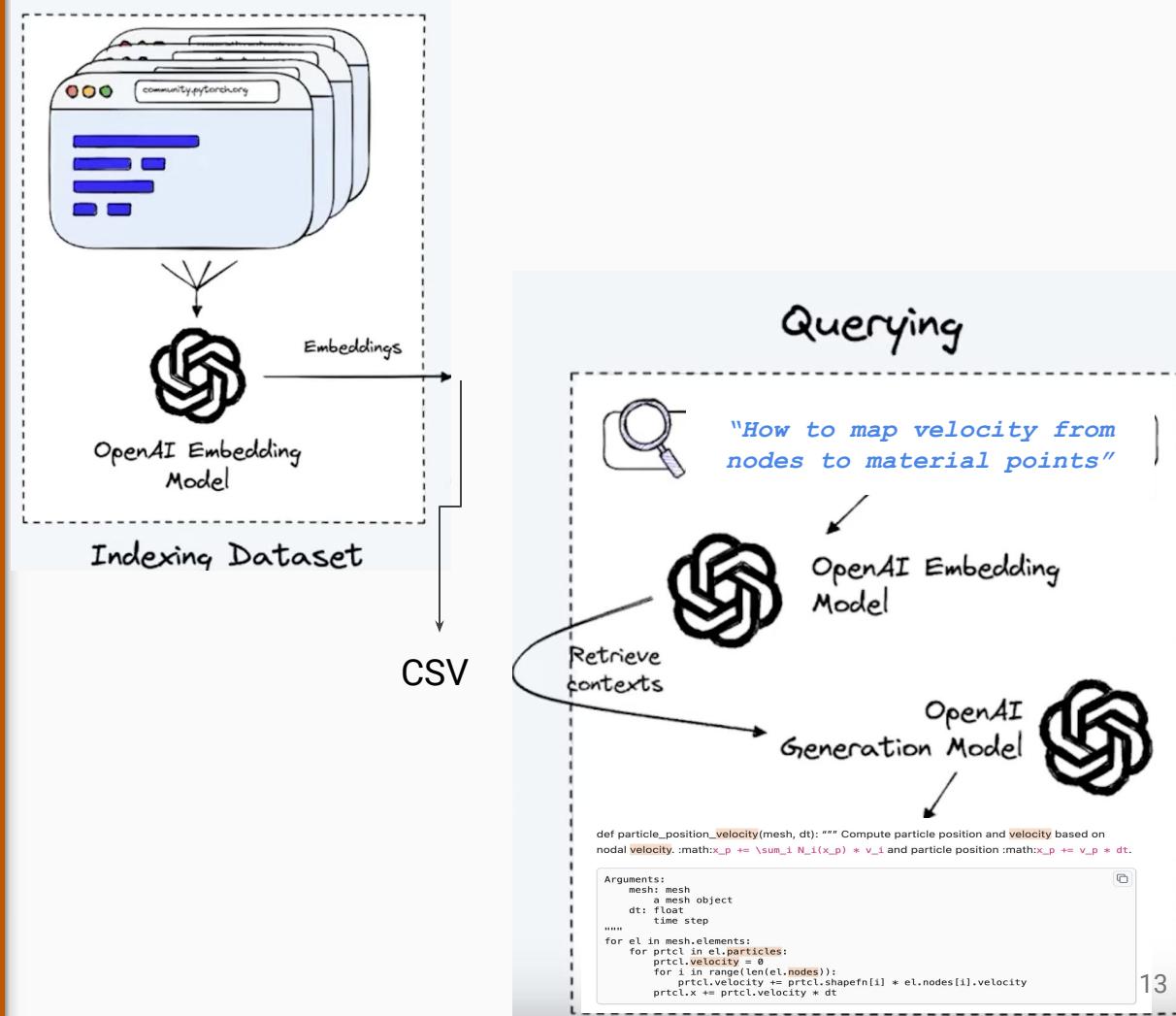
*"The **Eiffel Tower** is a **wrought iron lattice tower** on the Champ de Mars in Paris."*

If the query is “Landmarks in France” without using the words “Eiffel Tower” or “Wrought iron lattice tower”, we can never find the Eiffel Tower!



Building a semantic search with GPT

- Use GPT to document and test codes
- Embed text as vectors
- Convert search string to embedded vector
- Do a similarity search
- Provide solution with citation



Document generation with ChatGPT to provide “context”

K

Simplify the following Python script removing redundancies:

```
# Python 3.7                                Original code: Prof. Pedro Arduino (UW)
import os
cur_dir = os.getcwd()
if ('jupyter/MyData' in cur_dir ):
    cur_dir = cur_dir.split('MyData').pop()
    storage_id = 'designsafe.storage.default'
    input_dir = ag.profiles.get()['username']+ cur_dir
    input_uri = 'agave://{}{}'.format(storage_id,input_dir)
    input_uri = input_uri.replace(" ","%20")
elif('jupyter/mydata' in cur_dir ):
    cur_dir = cur_dir.split('mydata').pop()
    storage_id = 'designsafe.storage.default'
    input_dir = ag.profiles.get()['username']+ cur_dir
    input_uri = 'agave://{}{}'.format(storage_id,input_dir)
    input_uri = input_uri.replace(" ","%20")
elif('jupyter/MyProjects' in cur_dir):
    cur_dir = cur_dir.split('MyProjects/').pop()
    PRJ = cur_dir.split('/')[0]
    qq = {"value.projectId": str(PRJ)}
    cur_dir = cur_dir.split(PRJ).pop()
    project_uuid = ag.meta.listMetadata(q=str(qq))[0]["uuid"]
    input_dir = cur_dir
    input_uri = 'agave://project-{}{}'.format(project_uuid,cur_dir)
    input_uri = input_uri.replace(" ","%20")
elif('jupyter/projects' in cur_dir):
```



Copy code

```
import os
cur_dir = os.getcwd()

if 'jupyter/MyData' in cur_dir or 'jupyter/mydata' in cur_dir:
    cur_dir = cur_dir.split('MyData').pop() if 'MyData' in cur_dir else
    cur_dir.split('mydata').pop()
    storage_id = 'designsafe.storage.default'
    input_dir = ag.profiles.get()['username']+ cur_dir
    input_uri = 'agave://{}{}'.format(storage_id,input_dir)

elif 'jupyter/MyProjects' in cur_dir or 'jupyter/projects' in cur_dir:
    cur_dir = cur_dir.split('MyProjects/').pop() if 'MyProjects' in
    cur_dir else cur_dir.split('projects/').pop()
    PRJ = cur_dir.split('/')[0]
    qq = {"value.projectId": str(PRJ)}
    cur_dir = cur_dir.split(PRJ).pop()
    project_uuid = ag.meta.listMetadata(q=str(qq))[0]["uuid"]
    input_dir = cur_dir
    input_uri = 'agave://project-{}{}'.format(project_uuid,cur_dir)

elif 'jupyter/CommunityData' in cur_dir:
    cur_dir = cur_dir.split('jupyter/CommunityData').pop()
    input_dir = cur_dir
    input_uri = 'agave://designsafe.storage.community
    /{}'.format(input_dir)
```

ChatGPT for generating a function from a script

K

Convert the response to a function that takes the argument `cur_dir` and returns `inputs`:



Copy code

```
import os

def simplify_script(cur_dir):
    if 'jupyter/MyData' in cur_dir or 'jupyter/mydata' in cur_dir:
        cur_dir = cur_dir.split('MyData').pop() if 'MyData' in cur_dir
    else cur_dir.split('mydata').pop()
        storage_id = 'designsafe.storage.default'
        input_dir = ag.profiles.get()['username']+ cur_dir
        input_uri = 'agave://{}//{}'.format(storage_id,input_dir)

    elif 'jupyter/MyProjects' in cur_dir or 'jupyter/projects' in
cur_dir:
        cur_dir = cur_dir.split('MyProjects/').pop() if 'MyProjects' in
cur_dir else cur_dir.split('projects/').pop()
        PRJ = cur_dir.split('/')[-1]
        qq = {"value.projectId": str(PRJ)}
        cur_dir = cur_dir.split(PRJ).pop()
        project_uuid = ag.meta.listMetadata(q=str(qq))[0]["uuid"]
        input_dir = cur_dir
        input_uri = 'agave://project-{}//{}'.format(project_uuid,cur_dir)
```

Adding a testing case inside the function

K

test the function for possible errors and issues



Testing the function for possible errors and issues would require knowing the context in which the script is being used and the expected output. Here are a few things you could check for:

- Make sure that the `ag` variable is defined and has the required methods (``profiles.get()`` and ``meta.listMetadata()``) before calling the function
- Verify that the input directory passed to the function is a valid path.
- Verify that the input_uri returned by the function is a valid URI and it points to the correct location.
- Check that the returned inputs dictionary has the correct format and keys.
- Also, you should check if there is any other edge cases that are not handled by the function, like if the cur_dir variable has a different format than expected.
- You can check the function's behavior when the cur_dir variable is empty or None.
- You can also test the function by passing different cur_dir variable which will cover different cases mentioned in the if elif block.

Once you've tested the function thoroughly and are satisfied that it works as expected, you can use it in your script.

K

Add a test to see if cur_dir is empty or none in the function



Here is an example of how you can add a test to check if the `cur_dir` variable passed to the function is empty or None:

Copy code

```
import os

def simplify_script(cur_dir):
    if cur_dir is None or cur_dir.strip() == '':
        raise ValueError("Input directory is empty or None")
    if 'jupyter/MyData' in cur_dir or 'jupyter/mydata' in cur_dir:
        cur_dir = cur_dir.split('MyData').pop() if 'MyData' in cur_dir
```

Now, if the function is called with an empty or None cur_dir variable, it will raise a ValueError with the message "Input directory is empty or None". You can test this function by passing empty or None value to cur_dir variable, and check if it raises ValueError or not.

Copy code

```
try:
    current_directory = None
    inputs = simplify_script(current_directory)
except ValueError as e:
    print(e)
```

It should print "Input directory is empty or None".

Unit test and documentation

K

Write unit tests for the above function and add documentation as docstring



Here is an example of how you can write unit tests for the above function:

```
import unittest

class TestSimplifyScript(unittest.TestCase):

    def test_valid_input_dir(self):
        cur_dir = '/jupyter/MyData/myfolder'
        expected_inputs = {'inputDirectory':
['agave://designsafe.storage.default/username/myfolder']}
        inputs = simplify_script(cur_dir)
        self.assertDictEqual(inputs, expected_inputs)

    def test_valid_input_dir_MyProjects(self):
        cur_dir = '/jupyter/MyProjects/project1/myfolder'
        expected_inputs = {'inputDirectory': ['agave://project-
uuid/myfolder']}
        inputs = simplify_script(cur_dir)
        self.assertDictEqual(inputs, expected_inputs)
```

Copy code

You can run the test by running the code above, it will print the test results.

And here is an example of how you can add documentation as docstring to the function:

Copy code

```
import os

def simplify_script(cur_dir):
    """
    Function which will simplify the script by removing redundancies.
    It takes the argument cur_dir and returns inputs.
    It checks if the cur_dir is None or empty string, it will raise
    ValueError.

    :param cur_dir: current working directory
    :type cur_dir: str
    :return: inputs
    :rtype: dict
    :raises: ValueError
    """

    if cur_dir is None or cur_dir.strip() == '':
        raise ValueError("Input directory is empty or None")
    if 'jupyter/MyData' in cur_dir or 'jupyter/mydata' in cur_dir:
        cur_dir = cur_dir.split('MyData')[0] if 'MyData' in cur_dir
```

Testing on the CommunityData `LearnMPM`

The screenshot shows the DesignSafe-CI Data Depot interface. At the top, there is a navigation bar with links for django CMS, DesignSafe-CI, Page, Language, and Create. Below the navigation bar is the DesignSafe logo and social media icons. The main menu includes Workspace, Learning Center, NHERI Facilities, NHERI Community, News, Help, and a search bar. On the left, a sidebar lists categories like My Data, My Projects, Shared with Me, Box.com, Dropbox.com, Google Drive, Published, Published (NEES), and Community Data. The main content area is titled "DATA DEPOT" and shows a list of files under the "LearnMPM" folder. The table has columns for Name, Size, and Last modified. The "Name" column is checked.

Name	Size	Last modified
LearnMPM	--	1/3/23 12:57 PM
postprocess.ipynb	3.1 kB	5/19/22 2:18 PM
preprocess.ipynb	3.1 kB	5/19/22 2:18 PM

Auto-documenting LearnMPM code with GPT

```
import numpy as np

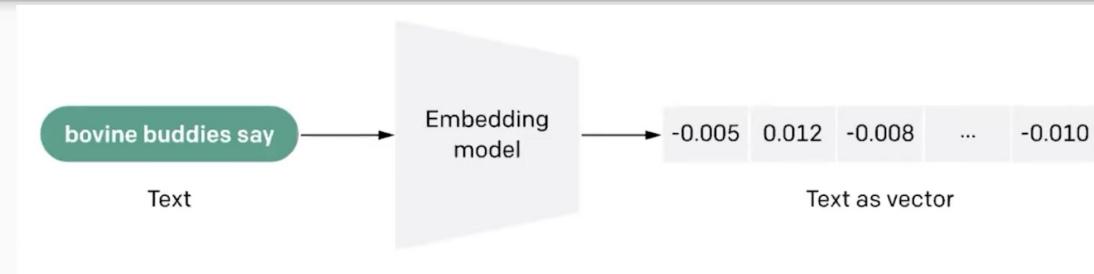
class LinearElastic1D:
    def __init__(self, E, density):
        self.E = E
        self.density = density

    def update_stress(self, particle):
        particle.stress+=particle.dstrain*self.E
```

Recognized “E” is the
Young’s modulus

```
1  import numpy as np
2
3  class LinearElastic1D:
4      """
5          Linear Elastic material for 1D
6
7          Attributes:
8
9          E: float
10         | Young's modulus
11
12         density: float
13         | Density of material point
14         """
15     def __init__(self, E, density):
16         """Initialize material properties
17
18         Args:
19             E: Young's modulus
20
21             density: Density
22             """
23
24         self.E = E
25         self.density = density
26
27     def update_stress(self, particle):
28         """Update particle stress based on particle strain.
29         :math:`\sigma = E \cdot \epsilon` and
30         the stress :math:`\sigma = \sigma + d\sigma`.
31
32         Args:
33             particle: Particle object
34             E: Young's modulus
35             """
36
37         particle.stress+=particle.dstrain*self.E
```

Text embedding: Extract function definitions as vectors



Extract everything starting with a `def`

```
import os
from glob import glob
import pandas as pd

def get_function_name(code):
    """
    Extract function name from a line beginning with "def "
    """
    assert code.startswith("def ")
    return code[len("def "):code.index("(")]

def get_until_no_space(all_lines, i) -> str:
    """
    Get all lines until a line outside the function definition is found.
    """
    ret = [all_lines[i]]
    for j in range(i + 1, i + 10000):
        if j < len(all_lines):
            if len(all_lines[j]) == 0 or all_lines[j][0] in [" ", "\t", ")"]:
                ret.append(all_lines[j])
            else:
                break
    return "\n".join(ret)

def get_functions(filepath):
    """
    Get all functions in a Python file.
    """
    pass
```

Total number of py files: 12
Total number of functions extracted: 21

Embed text as vectors

- Future store it in a Vector DB rather than a CSV file

```
from openai.embeddings_utils import get_embedding
import openai

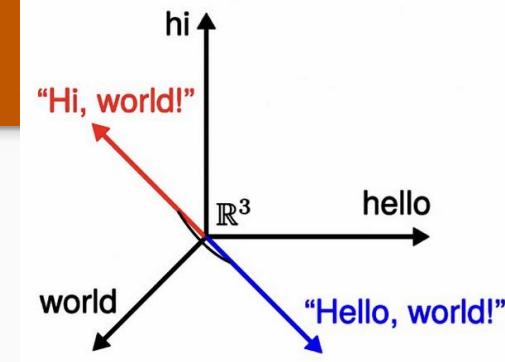
# Authenticate with OpenAI API
openai.api_key = "[API Key]"

df = pd.DataFrame(all_funcs)
df['code_embedding'] = df['code'].apply(lambda x: get_embedding(x, engine='text-embedding-ada-002'))
df['filepath'] = df['filepath'].apply(lambda x: x.replace(code_root, ""))
df.to_csv("data/code_search_openai-python.csv", index=False)
df.head()
```

	code	function_name	filepath	code_embedding
0	def nodal_total_force(mesh):\n """\n Com...	nodal_total_force	/update.py	[-0.008052323944866657, 0.025560934096574783, ...]
1	def nodal_acceleration_velocity(mesh, dt):\n ...	nodal_acceleration_velocity	/update.py	[0.008488386869430542, 0.024847134947776794, -...]
2	def nodal_velocity(mesh):\n """Compute noda...	nodal_velocity	/update.py	[-0.003850535023957491, 0.013598286546766758, ...]
3	def fix_nodal_bc_momentum(mesh):\n """Set m...	fix_nodal_bc_momentum	/update.py	[-0.01586133986711502, -0.005389683414250612, ...]
4	def fix_nodal_bc_force(mesh):\n """Set noda...	fix_nodal_bc_force	/update.py	[-0.01658795401453972, 0.004961120896041393, -.21

Cosine similarity

- Convert Query to Vector
- Run “cosine similarity”
- Fetch results and identify sources!



```
from openai.embeddings_utils import cosine_similarity

def search_functions(df, code_query, n=3, pprint=True, n_lines=7):
    embedding = get_embedding(code_query, engine='text-embedding-ada-002')
    df['similarities'] = df.code_embedding.apply(lambda x: cosine_similarity(x, embedding))

    res = df.sort_values('similarities', ascending=False).head(n)
    if pprint:
        for r in res.iterrows():
            print(r[1].filepath+":"+r[1].function_name + " score=" + str(round(r[1].similarities, 3)))
            print("\n".join(r[1].code.split("\n")[:n_lines]))
            print('*'*70)
    return res
```

Semantic search of the codebase

"How do I map velocity from nodes to material points?"

```
res = search_functions(df, 'How do I map velocity from nodes to material points?', n=3)
```

```
/update.py:particle_position_velocity score=0.757
def particle_position_velocity(mesh, dt):
    """
    Compute particle position and velocity based on nodal velocity. :math:`x_p += \sum_i N_i(x_p) * v_i`
    Arguments:
        mesh: mesh
            a mesh object
    """

Arguments:
```

```
/update.py:particle_velocity score=0.75
def particle_velocity(mesh, dt):
    """
    Compute particle velocity transfer nodal velocity to particle. :math:`v_p += \sum_i N_i(x_p) * f_i(t)`
    Arguments:
        mesh: mesh
            a mesh object
    """

Arguments:
```

```
/update.py:nodal_velocity score=0.75
def nodal_velocity(mesh):
    """Compute nodal velocity as :math:`v = mv / m`."""
    Arguments:
        mesh: mesh
            a mesh object
    """

Arguments:
```

Cites its sources!

Found the right function!!! Yay :)

```
def particle_position_velocity(mesh, dt): """ Compute particle position and velocity based on
nodal velocity. :math:x_p += \sum_i N_i(x_p) * v_i and particle position :math:x_p += v_p * dt.
```

```
Arguments:
    mesh: mesh
        a mesh object
    dt: float
        time step
    """
    for el in mesh.elements:
        for prtcl in el.particles:
            prtcl.velocity = 0
            for i in range(len(el.nodes)):
                prtcl.velocity += prtcl.shapefn[i] * el.nodes[i].velocity
            prtcl.x += prtcl.velocity * dt
```

Adding a frontend with Plotly Dash

localhost:8050

DesignSafe Semantic Search Engine for Python Code

From nodes to material points?

/update.py:particle_position_velocity score=0.757

```
def particle_position_velocity(mesh, dt): """ Compute particle position and velocity based on nodal velocity. :math:`x_p += \sum_i N_i(x_p) * v_i` and particle position :math:`x_p += v_p * dt`. Arguments: mesh: mesh a mesh object dt: float time step """ for el in mesh.elements: for prtcl in el.particles: prtcl.velocity = 0 for i in range(len(el.nodes)): prtcl.velocity += prtcl.shapefn[i] * el.nodes[i].velocity prtcl.x += prtcl.velocity * dt
```

/update.py:particle_velocity score=0.75

```
def particle_velocity(mesh, dt): """ Compute particle velocity transfer nodal velocity to particle. :math:`v_p += \sum_i N_i(x_p) * f_{total} / m_i * dt`. Arguments: mesh: mesh a mesh object dt: float time step """ for el in mesh.elements: for prtcl in el.particles: for i in range(el.nnodes): prtcl.velocity += el.nodes[i].f_total/el.nodes[i].mass * prtcl.shapefn[i] * dt
```

/update.py:nodal_velocity score=0.75

```
def nodal_velocity(mesh): """Compute nodal velocity as :math:`v = mv / m`. Arguments: mesh: mesh a mesh object """ for node in mesh.nodes: if(node.mass > 0.): node.velocity = node.momentum / node.mass
```

/update.py:momentum_in_nodes score=0.743

```
def momentum_in_nodes(mesh, dt): """Compute momentum in nodes. :math:`mv += force * dt` Arguments: mesh: mesh a mesh object dt: float time step """ for node in mesh.nodes: node.momentum += node.f_total * dt
```

/interpolate.py:mass_momentum_to_nodes score=0.738

```
def mass_momentum_to_nodes(mesh): """Map mass and momentum of particles in the elements to the associated nodes. Args: mesh: mesh object with nodes, elements and particles """ for el in mesh.elements: for par in el.particles: for i in range(len(el.nodes)): el.nodes[i].mass += par.mass * par.shapefn[i] el.nodes[i].momentum += par.mass * par.shapefn[i] * par.velocity
```

Integrate directly in DesignSafe Jupyter

The screenshot shows a Jupyter Notebook interface with the URL https://jupyter.designsafe-ci.org/user/kks32/lab/tree/MyData/uniaxial_stress/semantic-search.ipynb. The notebook has three tabs: 'jupyter@jupyter-kks32: ~' (active), 'mpmipy.ipynb', and 'semantic-search.ipynb'. The 'semantic-search.ipynb' tab displays the following code:

```
[12]: res = search_functions(df, 'How do I map velocity from nodes to material points?', n=3, n_lines=10)

/update.py:particle_position_velocity score=0.757
def particle_position_velocity(mesh, dt):
    """
    Compute particle position and velocity based on nodal velocity. :math:`x_p += \sum_i N_i(x_p) * v_i` and particle
    position :math:`x_p += v_p * dt`.

    Arguments:
        mesh: mesh
            a mesh object
        dt: float
            time step
    """

/update.py:particle_velocity score=0.75
def particle_velocity(mesh, dt):
    """
    Compute particle velocity transfer nodal velocity to particle. :math:`v_p += \sum_i N_i(x_p) * {f_{total}}_i / m_i
    * dt`.

    Arguments:
        mesh: mesh
            a mesh object
        dt: float
            time step
    """

/update.py:nodal_velocity score=0.75
def nodal_velocity(mesh):
    """
    Compute nodal velocity as :math:`v = mv / m`.
    """

Arguments:
    mesh: mesh
        a mesh object
    """
```

The left sidebar shows a file tree with files like 'code_sear...', 'entity_set...', 'mesh.txt', 'mpm.ipynb', 'mpm.json', 'mpmipy.ipynb', 'particles.txt', 'results.ipynb', 'semantic-search.ipynb' (selected), and 'test_benc...'. The bottom status bar indicates 'Mode: Command' and 'Ln 1, Col 1 semantic-search.ipynb'.

Searching Docs

Can we provide context to doc search?

Original GPT without context - telling lies as it invents a new Tapipy website and App to generate a token

```
▶ 
    prompt = "How to generate a token using Tapipy"

    openai.Completion.create(prompt=prompt, temperature=0, max_tokens=300, model=COMPLETIONS_MODEL)["choices"][0]["text"].strip(" \n")
[20] ✓ 10.4s
Python
...
1. Create an account on Tapipy.\n\n2. Log in to your account and go to the "My Apps" page.\n\n3. Click on "Create New App" and enter the details of your app.\n\n4. Once your app is created, click on "Generate Token".\n\n5. Enter the details of the token you want to generate and click "Generate".\n\n6. Your token will be generated and displayed on the screen.'
```

When you ask a question for which it can find a context! - It answers correctly!

```
[17] ✓ 2.7s
Python
answer_query_with_context("How to generate a token using Tapipy", df)
...
'Use the t.gettokens() command to generate a token using Tapipy.'
```

When it doesn't know...at least it is honest!

```
[18] ✓ 0.9s
Python
answer_query_with_context("How to access files using Tapipy", df)
...
'"I don't know."
```

How to provide context

Providing Context to GPT

What if we could provide GPT with some context so it can provide useful help!

+ Code + Markdown

```
prompt = """Answer the question as truthfully as possible, and if you're unsure of the answer, say "Sorry, I don't know".
```

Context:

Create an Tapis Client Object

The first step in using the Tapis Python SDK, tapipy, is to create a Tapis Client object. First, import the Tapis class and create python object called t

```
# Import the Tapis object
from tapipy.tapis import Tapis
```

```
# Log into you the Tapis service by providing user/pass and url.
t = Tapis(base_url='https://tacc.tapis.io',
           username='your username',
           password='your password')
```

Generate a Token

With the t object instantiated, we can exchange our credentials for an access token. In Tapis, you never send your username and password directly to the se

```
# Get tokens that will be used for authenticated function calls
t.get_tokens()
print(t.access_token.access_token)
```

```
Out[1]: eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9...
```

Note that the tapipy t object will store and pass your access token automatically.

Q: How to generate a token using Tapipy?

A:

```
Q: How to generate a token using Tapipy?
A:
"""
openai.Completion.create(
    prompt=prompt,
    temperature=0,
    max_tokens=300,
    model=COMPLETIONS_MODEL
)["choices"][0]["text"].strip("\n")
```

Python

To generate a token using Tapipy, first import the Tapis class and create a Python object called t that points to the Tapis server using your TACC username and password. Then, use the t object to get tokens that will be used for authenticated function calls by calling the get_tokens() method. Finally, print the access token to use for authentication.'

Future work

- Expand to all DesignSafe Data Depot codes
- Language specific search
- Integration with Jupyter
- Semantic search to include documentation
- Store embedded DB in a Vector DB

