SymSolver: gitlab.com/Sevans7/symsolver

Samuel Evans · Boston University PhD Candidate · sevans7@bu.edu

Example: Acoustic Waves

```python
import SymSolver.expressions as xp

## DEFINE SYMBOLS ##
n, P = xp.Symbols(['n', 'P'])
u     = xp.Symbol('u', vector=True)
m     = xp.Symbol('m', constant=True)
gamma = xp.Symbol(r'\gamma', constant=True)

## PUT SYMBOLS TOGETHER TO MAKE EQUATIONS ##
continuity = xp.Equation(n.ddt() + n * u.div(), 0)
momentum   = xp.Equation(n * u.ddt(), -1 / m * P.grad())
heating    = xp.Equation(P.ddt() + gamma * P * u.div(), 0)

## PUT EQUATIONS TOGETHER TO MAKE AN EQUATIONSYSTEM ##
eqs = xp.EquationSystem(continuity, momentum, heating)

## SHOW OBJECT ##
eqs.view()
```

output >>

$$
\begin{aligned}
\frac{\partial}{\partial t} n + \vec{u} \cdot \nabla n + n \left[\nabla \cdot \vec{u} \right] &= 0 \\
n \left(\frac{\partial}{\partial t} \vec{u} + \left(\vec{u} \cdot \nabla \right) \vec{u} \right) &= - \frac{\left[\nabla P \right]}{m} \\
\frac{\partial}{\partial t} P + \vec{u} \cdot \nabla P + \gamma P \left[\nabla \cdot \vec{u} \right] &= 0
\end{aligned}
$$

```python
## GET LATEX COPY-PASTA ##
print(eqs)
```

```
\begin{align}
  \frac{\partial}{\partial t} n + \vec{u} \cdot \nabla n + n \left[ \nabla \cdot  \vec{u} \right] &= 0 \\
  n \left( \frac{\partial}{\partial t} \vec{u} + \left( \vec{u} \cdot \nabla \right) \vec{u} \right) &= -  \frac{\left[ \nabla P \right]}{m} \\
  \frac{\partial}{\partial t} P + \vec{u} \cdot \nabla P + \gamma P \left[ \nabla \cdot  \vec{u} \right] &= 0
\end{align}
```

```python
## LINEARIZE ##
eqs = eqs.linearize()

## ASSUME PLANE WAVES ##
eqs = eqs.assume_plane_waves()

# ASSUME CONSTANT BACKGROUND
eqs = eqs.assume_o0_constant()
# ASSUME u0 == 0  (for simplicity)
eqs = eqs.substitute([u.o0(), 0])

## SIMPLIFY ##
eqs = eqs.simplify()
eqs.view(index_from=0)
```

output >>

$$
\begin{aligned}
&(0) \quad & -i\omega n^{(1)} + i n^{(0)} \left(\vec{k} \cdot \vec{u}^{(1)} \right) &= 0 \\
&(1) \quad & n^{(0)} \omega \vec{u}^{(1)} &= \frac{\vec{k} P^{(1)}}{m} \\
&(2) \quad & -i\omega P^{(1)} + i\gamma P^{(0)} \left(\vec{k} \cdot \vec{u}^{(1)} \right) &= 0
\end{aligned}
$$

SymSolver: gitlab.com/Sevans7/symsolver

Samuel Evans · Boston University PhD Candidate · sevans7@bu.edu

Example: Acoustic Waves

$$(0) \qquad -i\omega n^{(1)} + i n^{(0)} \left(\vec{k} \cdot \vec{u}^{(1)}\right) = 0$$

$$(1) \qquad n^{(0)} \omega \vec{u}^{(1)} = \frac{\vec{k} P^{(1)}}{m}$$

$$(2) \qquad -i\omega P^{(1)} + i\gamma P^{(0)} \left(\vec{k} \cdot \vec{u}^{(1)}\right) = 0$$

<< value of eqs <<

```python
## SOLVE ##
disprel = eqs.o1solve(show_work=True)
disprel.view()
```

output >>

$$-i\omega + i\frac{\gamma P^{(0)}\left(\vec{k} \cdot \vec{k}\right)}{m n^{(0)} \omega} = 0$$

```python
## CONVERT TO "POLYNOMIAL" FORM ##
disprel.polynomialize(omega).view()
```

output >>

$$-i\omega^2 + i\frac{P^{(0)}\gamma\left(\vec{k} \cdot \vec{k}\right)}{n^{(0)} m} = 0$$

Theory ("by hand")

$$\omega^2 = \frac{P^{(0)}\gamma k^2}{n^{(0)} m}$$

```
Solving equation (1) for \vec{u}^{(1)} (then simplifying equation (1)):
```

$$\vec{u}^{(1)} = \frac{\vec{k} P^{(1)}}{m n^{(0)} \omega}$$

```
Plugging into equation (0) the value from equation (1) for \vec{u}^{(1)}
Then simplifying equation (0)
Plugging into equation (2) the value from equation (1) for \vec{u}^{(1)}
Then simplifying equation (2)
--- The updated list of equations is ---
```

$$(0) \qquad -i\omega n^{(1)} + i\frac{\left(\vec{k} \cdot \vec{k}\right) P^{(1)}}{m\omega} = 0$$

$$(2) \qquad \left(-i\omega + i\frac{\gamma P^{(0)}\left(\vec{k} \cdot \vec{k}\right)}{m n^{(0)} \omega}\right) P^{(1)} = 0$$

```
Equation looks like (A * (P^{(1)}) == 0). This implies (A == 0)
Getting the dispersion relation from equation (2):
```

$$-i\omega + i\frac{\gamma P^{(0)}\left(\vec{k} \cdot \vec{k}\right)}{m n^{(0)} \omega} = 0$$

```
--- The updated list of equations is ---
```

$$(0) \qquad -i\omega n^{(1)} + i\frac{\left(\vec{k} \cdot \vec{k}\right) P^{(1)}}{m\omega} = 0$$

```
/Users/Sevans/Code/SymSolver/SymSolver/linearizing.py:597: UserWarning: E
quationSystem (id=<0x120692a00>) contains unnecessary equations. I found
the dispersion relation without ever using equation(s): [0]
  warnings.warn(extra_info_warning)
```

Also supports (not shown in this talk):

- **Subscripts, Superscripts, Summation notation**
- **Numerical substitutions, Numpy array values**
- **Solving numerically (with multiprocessing if arrays)**
- **Cross products, Curl, Vector components**