Dimensional analysis in UFL

Part 2/2

Andreas Zilian, Michal Habera

Department of Engineering | University of Luxembourg

20th June 2025 | FEniCS 2025

UNIVERSITÉ DU LUXEMBOURG

Contents

Recall | from earlier talk *Physical units in UFL*

We introduced

- class `Quantity` to represent physical quantities with units,
  ```
  dolfiny.units.Quantity(mesh, scale, unit, symbol)
  ```
- transformation of expressions and forms,
  ```
  dolfiny.units.transform(expr, mapping)
  ```
- factorization of expressions and forms ("check" or "factorize" modes),
  ```
  dolfiny.units.factorize(expr, quantities, mapping, mode)
  ```

Motivation | why non-dimensionalisation?

1. Inspect relative scales of terms in expressions/forms.
   $\longrightarrow$ identify dominant and (potentially) negligible terms
2. Improve numerical conditioning.
   $\longrightarrow$ avoid issues with vastly different scales
3. Derive dimensionless numbers.
   $\longrightarrow$ discover fundamental parameters (Reynolds, Euler, Strouhal, …)
4. Simplify parameter studies.
   $\longrightarrow$ reduce number of independent variables
5. Enable universal solutions.
   $\longrightarrow$ results applicable across different physical scales

Contents

Buckingham Pi theorem

- a general, abstract method for dimensional analysis of a set of dimensional quantities,
- useful even without a specific PDE at hand,
- early detection of dimensional inconsistencies and dependent phenomena.

Figure: *Edgar Buckingham (1867-1940)*

Buckingham π theorem | formal statement

If we have a physical system described by some

$$f(q_1, q_2, \ldots, q_n) = 0,$$

where q_i are n dimensional quantities, then there exist p dimensionless quantities π_i (so-called π-groups) such that

$$F(\pi_1, \pi_2, \ldots, \pi_p) = 0.$$

Moreover,

$$p = \dim \ker \mathbf{M},$$

where $\mathbf{M} \in \mathbb{Q}^{k \times n}$ is called the *dimensional matrix* where M_{ij} represents the exponent of the i-th dimension in parameter q_j.

Lay words: There are as many dimensionless numbers, as there are independent ways of combining the dimensional quantities.

Buckingham π theorem | example

Flow around a sphere

Consider the drag force F_D on a sphere of diameter d moving through a fluid with velocity v, density ρ, and dynamic viscosity μ. The relevant parameters are $\{F_D, d, v, \rho, \mu\}$ with dimensions

$$[F_D] = \mathsf{LMT}^{-2}, \quad [d] = \mathsf{L}, \quad [v] = \mathsf{LT}^{-1}, \quad [\rho] = \mathsf{L}^{-3}\mathsf{M}, \quad [\mu] = \mathsf{L}^{-1}\mathsf{MT}^{-1}.$$

The *dimensional matrix* is

$$\mathbf{M} = \begin{array}{|c|c|c|c|c|} \hline 1 & 1 & 1 & -3 & -1 \\ \hline 1 & 0 & 0 & 1 & 1 \\ \hline -2 & 0 & -1 & 0 & -1 \\ \hline \end{array} \qquad \text{nullspace}(\mathbf{M}) = \begin{array}{|c|c|c|c|c|} \hline 1 & -2 & -2 & -1 & 0 \\ \hline 0 & 1 & 1 & 1 & -1 \\ \hline \end{array}$$

The *kernel dimension* is $p = 2$ (two independent dimensionless groups)

$$\Pi_1 = \frac{F_D}{\rho v^2 d^2} \qquad \text{(drag coefficient, } C_D\text{)},$$

$$\Pi_2 = \frac{\rho v d}{\mu} \qquad \text{(Reynolds number, Re)}.$$

Contents

Non-dimensionalisation | general considerations

- Forms (describing a multi-field PDE problem) can carry different units
- Adding forms should be done on their non-dimensional equivalents
- Approach non-dimensionalisation for each form separately

Non-dimensionalisation | procedure (per form)

1. Establish dimensional quantities (parameters and reference scales)

```
from dolfiny.units import Quantity
k = Quantity(mesh, 1, syu.kilo * syu.newton / syu.meter, "k")
f = Quantity(mesh, 1, syu.pascal, "f")
u_ref = Quantity(mesh, 1, syu.millimeter, "u_ref")
l_ref = Quantity(mesh, 1, syu.meter, "l_ref")
quantities = [k, f, u_ref, l_ref]
```

2. Identify dimensionless π-groups

```
dolfiny.units.buckingham_pi_analysis(quantities)
```

3. Establish dimensional mapping of mesh length scale and function scales

```
mapping = {mesh.ufl_domain(): l_ref, u: u_ref * u, δu: u_ref * δu}
```

4. Split form into *relevant* named terms

```
terms = {"int": -ufl.inner(ufl.grad(δu), k * ufl.grad(u)) * dx,
         "ext": δu * f * dx}
```

5. Separate dimensional and non-dimensional factors in form terms

```
terms_fact = dolfiny.units.factorize(terms, quantities, mapping=mapping)
```

6. Normalise factorised form terms with *selected* reference term

```
terms_norm = dolfiny.units.normalize(terms_fact, "int", quantities)
```

7. Produce non-dimensional form

```
form_nondimensional = sum(terms_norm.values())
```

Contents

Example | incompressible Navier-Stokes

Weak form

$$0 = \underbrace{\int_\Omega \delta\mathbf{v} \cdot \rho \frac{\partial \mathbf{v}}{\partial t}\,\mathrm{d}x}_{\text{unsteady}} + \underbrace{\int_\Omega \delta\mathbf{v} \cdot \rho\,(\mathbf{v} \cdot \nabla\mathbf{v})\;\mathrm{d}x}_{\text{convection}} + \underbrace{\int_\Omega \mathbf{D}(\delta\mathbf{v}) : 2\rho\nu\,\mathbf{D}(\mathbf{v})\,\mathrm{d}x}_{\text{viscous}}$$

$$- \underbrace{\int_\Omega \nabla \cdot \delta\mathbf{v}\,p\,\mathrm{d}x}_{\text{pressure}} - \underbrace{\int_\Omega \delta\mathbf{v} \cdot \rho\mathbf{g}\,\mathrm{d}x}_{\text{force}} + \underbrace{\int_\Omega \delta p\,\nabla \cdot \mathbf{v}\,\mathrm{d}x}_{\text{incompressibility}} \qquad \forall\,(\delta\mathbf{v}, \delta p)$$

Quantities

$\nu = 1000\,\mathrm{mm}^2/\mathrm{s}$ $\quad \ell_{\mathrm{ref}} = 1\,\mathrm{m}$ $\quad v_{\mathrm{ref}} = 1\,\mathrm{m/s}$ $\quad g_{\mathrm{ref}} = 10\,\mathrm{m/s}^2$

$\rho = 5000\,\mathrm{kg/m}^3$ $\quad t_{\mathrm{ref}} = 1\,\mathrm{s}$ $\quad p_{\mathrm{ref}} = 5000\,\mathrm{Pa}$

Example | incompressible Navier-Stokes 2

Units implementation sketch

```
nu = Quantity(mesh, 1000, syu.millimeter**2 / syu.second, "nu")
rho = Quantity(mesh, 5000, syu.kilogram / syu.meter**3, "rho")
l_ref = Quantity(mesh, 1, syu.meter, "l_ref")
t_ref = Quantity(mesh, 1 / 60, syu.minute, "t_ref")
v_ref = Quantity(mesh, 1, syu.meter / syu.second, "v_ref")
p_ref = Quantity(mesh, 5000, syu.pascal, "p_ref")
g_ref = Quantity(mesh, 10, syu.meter / syu.second**2, "g_ref")
quantities = [nu, rho, l_ref, t_ref, v_ref, p_ref, g_ref]
```

```
dolfiny.units.buckingham_pi_analysis(quantities)
```

```
mapping = {mesh.ufl_domain(): l_ref, v: v_ref * v, p: p_ref * p, ...}
```

```
terms = {
    "unsteady": ufl.inner(δv, rho * (v - v0) / (t_ref * n)) * ufl.dx,
    "convection": ufl.inner(δv, rho * ufl.dot(v, ufl.grad(v))) * ufl.dx,
    "viscous": ufl.inner(D(δv), 2 * rho * nu * D(v)) * ufl.dx,
    "pressure": -ufl.inner(ufl.div(δv), p) * ufl.dx,
    "force": -ufl.inner(δv, rho * g_ref * b) * ufl.dx,
    "incompressibility": δp * ufl.div(v) * ufl.dx,
}
```

```
terms_fact = dolfiny.units.factorize(terms, quantities, mapping=mapping)
```

```
terms_norm = dolfiny.units.normalize(terms_fact, "convection", quantities)
```

```
form_nondimensional = sum(terms_norm.values())
```

Buckingham Pi analysis

```
==================================================
Buckingham Pi Analysis
==================================================
Symbol | Expression                    | Value (in base units)
-------+-----------------------------+----------------------------------
nu     | 1000.0*millimeter**2/second | 0.001*meter**2/second
rho    | 5000.0*kilogram/meter**3    | 5000.0*kilogram/meter**3
l_ref  | meter                       | meter
t_ref  | 0.01667*minute              | 1.0*second
v_ref  | meter/second                | meter/second
p_ref  | 5000.0*pascal               | 5000.0*kilogram/(meter*second**2)
g_ref  | 10.0*meter/second**2        | 10.0*meter/second**2

Dimension matrix (7 × 7):
Dimension           | nu | rho | l_ref | t_ref | v_ref | p_ref | g_ref
--------------------+----+-----+-------+-------+-------+-------+------
amount_of_substance | 0  | 0   | 0     | 0     | 0     | 0     | 0
current             | 0  | 0   | 0     | 0     | 0     | 0     | 0
length              | 2  | -3  | 1     | 0     | 1     | -1    | 1
luminous_intensity  | 0  | 0   | 0     | 0     | 0     | 0     | 0
mass                | 0  | 1   | 0     | 0     | 0     | 1     | 0
temperature         | 0  | 0   | 0     | 0     | 0     | 0     | 0
time                | -1 | 0   | 0     | 1     | -1    | -2    | -2

Dimensionless groups (4):
Group | Expression                  | Value
------+---------------------------+------
```

`Pi_1  | nu*t_ref/l_ref**2           | 0.001` $\mathrm{Re} = \Pi_2 = 1000$

`Pi_2  | l_ref*v_ref/nu              | 1e+03` $\mathrm{Fr} = \Pi_2 \Pi_4^{-1/2} = 10$

`Pi_3  | l_ref**2*p_ref/(nu**2*rho)  | 1e+06` $\mathrm{Eu} = \Pi_2^2 \Pi_3 = 1$

`Pi_4  | g_ref*l_ref**3/nu**2        | 1e+07` $\mathrm{St} = \Pi_1 \Pi_2 = 1$

Example | incompressible Navier-Stokes 4

Form normalisation

```
==================================================
Terms after normalization with "convection"
==================================================
Reference factor from 'convection':
Term       | Factor              | Value (in base units)
-----------+---------------------+------------------------------
convection | l_ref*rho*v_ref**3 | 5000.0*kilogram*meter/second**3

Term              | Factor                | Value (in base units)
------------------+-----------------------+----------------------
unsteady          | l_ref/(t_ref*v_ref)   | 1.000
convection        | 1                     | 1.000
viscous           | nu/(l_ref*v_ref)      | 0.001000
pressure          | p_ref/(rho*v_ref**2) | 1.000
force             | g_ref*l_ref/v_ref**2 | 10.00
incompressibility | p_ref/(rho*v_ref**2) | 1.000
==================================================
```

Example | neo-Hooke hyperelasticity

Weak form

$$0 = \underbrace{\int_\Omega \mathbf{C}(\delta\mathbf{u}) : \frac{\partial W_{\text{shear}}}{\partial \mathbf{C}(\mathbf{u})} \,\mathrm{d}x}_{\text{shear}} + \underbrace{\int_\Omega \mathbf{C}(\delta\mathbf{u}) : \frac{\partial W_{\text{bulk}}}{\partial \mathbf{C}(\mathbf{u})} \,\mathrm{d}x}_{\text{bulk}} - \underbrace{\int_\Gamma \delta\mathbf{u} \cdot \mathbf{t} \,\mathrm{d}s}_{\text{external}} \qquad \forall\, (\delta\mathbf{u})$$

with $\quad W(\mathbf{C}) = W_{\text{shear}} + W_{\text{bulk}} = \frac{\mu}{2}\,(I_1 - 3 - 2\ln J) + \frac{\kappa}{2}\,(J-1)^2$

Quantities

$\mu = 0.357\,\text{MPa} \qquad \ell_{\text{ref}} = 0.1\,\text{m}$

$\kappa = 1.667\,\text{MPa} \qquad t_{\text{ref}} = 1.0\,\text{MPa}$

Units implementation sketch

```
μ = Quantity(mesh, 0.357, syu.mega * syu.pascal, "μ")
κ = Quantity(mesh, 1.667, syu.mega * syu.pascal, "κ")
l_ref = Quantity(mesh, 0.1, syu.meter, "l_ref")
t_ref = Quantity(mesh, 1.0, syu.mega * syu.pascal, "t_ref")
quantities = [μ, κ, l_ref, t_ref]
```

```
dolfiny.units.buckingham_pi_analysis(quantities)
```

```
mapping = {mesh.ufl_domain(): l_ref, u: l_ref * u, δu: l_ref * δu}
```

```
terms = {
      "int_shear": ufl.inner(δC, ufl.diff(W_shear(C), C)) * dx,
      "int_bulk": ufl.inner(δC, ufl.diff(W_bulk(C), C)) * dx,
      "external": -ufl.inner(δu, t) * ds(surface_upper),
}
```

```
terms_fact = dolfiny.units.factorize(terms, quantities, mapping=mapping)
```

```
terms_norm = dolfiny.units.normalize(terms_fact, "int_bulk", quantities)
```

```
form_nondimensional = sum(terms_norm.values())
```

Buckingham Pi analysis

```
==================================================
Buckingham Pi Analysis
==================================================
Symbol | Expression       | Value (in base units)
-------+------------------+-----------------------------------
μ      | 3.571e+5*pascal | 3.571e+5*kilogram/(meter*second**2)
κ      | 1.667e+6*pascal | 1.667e+6*kilogram/(meter*second**2)
l_ref  | 0.1*meter        | 0.1*meter
t_ref  | 1.0e+6*pascal   | 1.0e+6*kilogram/(meter*second**2)

Dimension matrix (7 × 4):
Dimension            | μ  | κ  | l_ref | t_ref
---------------------+----+----+-------+------
amount_of_substance | 0  | 0  | 0     | 0
current              | 0  | 0  | 0     | 0
length               | -1 | -1 | 1     | -1
luminous_intensity  | 0  | 0  | 0     | 0
mass                 | 1  | 1  | 0     | 1
temperature          | 0  | 0  | 0     | 0
time                 | -2 | -2 | 0     | -2

Dimensionless groups (2):
Group | Expression | Value
------+------------+------
Pi_1  | κ/μ        | 4.67   incompressibility ratio or bulk-to-shear ratio
Pi_2  | t_ref/μ    | 2.8    relative magnitude of deformation
==================================================
```


UNIVERSITÉ DU LUXEMBOURG

Example | neo-Hooke hyperelasticity 4

Form normalisation

```
==================================================
Terms after normalization with "int_bulk"
==================================================
Reference factor from 'int_bulk':
Term     | Factor     | Value (in base units)
---------+------------+----------------------------------
int_bulk | l_ref**3*κ | 1667.0*kilogram*meter**2/second**2

Term      | Factor   | Value (in base units)
----------+----------+----------------------
int_bulk  | 1        | 1.000
int_shear | μ/κ      | 0.2143
external  | t_ref/κ  | 0.6000
==================================================
```

Example | electrodiffusion (Poisson-Nernst-Planck)

Weak forms

$$0 = \underbrace{\int_\Omega \nabla\delta\varphi \cdot \varepsilon_r\varepsilon_0 \nabla\varphi \,\mathrm{d}x}_{\text{potential}} - \underbrace{\int_\Omega \delta\varphi\, F\left(\sum_k z_k c_k + w\right) \mathrm{d}x}_{\text{electroneutrality}} \qquad \forall\,(\delta\varphi)$$

$$0 = \underbrace{\int_\Omega \sum_k \nabla\delta c_k \cdot D_k \nabla c_k \,\mathrm{d}x}_{\text{diffusion}} + \underbrace{\int_\Omega \sum_k \nabla\delta c_k \cdot D_k c_k z_k \frac{F}{RT} \nabla\varphi \,\mathrm{d}x}_{\text{convection}}$$

$$+ \underbrace{\int_\Omega \sum_k \nabla\delta c_k \cdot D_k c_k \nabla a \,\mathrm{d}x}_{\text{Debye expansion term 0-th}} + \underbrace{\int_\Omega \sum_k \nabla\delta c_k \cdot D_k c_k \nabla(ab) \,\mathrm{d}x}_{\text{Debye expansion term 1-st}} \qquad \forall\,(\delta c_k)$$

Quantities

$$\varepsilon_0 = 8.854 \times 10^{-12}\ \mathrm{F/m} \qquad e_0 = 1.602 \times 10^{-19}\ \mathrm{C} \qquad \varphi_{\mathrm{ref}} = 1\ \mathrm{V}$$

$$F = 9.649 \times 10^{4}\ \mathrm{C/mol} \qquad R = 8.314\ \mathrm{J/(K\,mol)} \qquad c_{\mathrm{ref}} = 50\ \mathrm{mol/m^3}$$

$$D_{\mathrm{ref}} = 1.0 \times 10^{-10}\ \mathrm{m^2/s} \qquad T = 300\ \mathrm{K} \qquad l_{\mathrm{ref}} = 1\ \text{Å}$$

Example | electrodiffusion (Poisson-Nernst-Planck) 2

Units implementation sketch

```
ε_0 = Quantity(mesh, 8.854187819e-12, syu.farad / syu.meter, "ε_0")
e_0 = Quantity(mesh, 1.602176634e-19, syu.coulomb, "e_0")
F = Quantity(mesh, 9.648533212e04, syu.coulomb / syu.mol, "F")
R = Quantity(mesh, 8.314462618, syu.joule / syu.kelvin / syu.mol, "R")
T = Quantity(mesh, 300.0, syu.kelvin, "T")
c_ref = Quantity(mesh, 50.0, syu.mol / syu.meter**3, "c_ref")
φ_ref = Quantity(mesh, 1.0, syu.volt, "φ_ref")
l_ref = Quantity(mesh, 1, syu.angstrom, "l_ref")
D_ref = Quantity(mesh, 1.0e-10, syu.meter**2 / syu.second, "D_ref")
quantities = [c_ref, φ_ref, D_ref, ε_0, F, R, T, l_ref, e_0]
```

```
dolfiny.units.buckingham_pi_analysis(quantities)
```

```
mapping = {mesh.ufl_domain(): l_ref, c: c_ref * c, φ: φ_ref * φ, ...}
```

```
terms_φ = {
    "potential": ..., "electroneutrality": ...,
}
terms_c = {
    "diffusion": ..., "convection": ..., "debye_0th": ..., "debye_1st": ...,
}
```

```
terms_φ_fact = dolfiny.units.factorize(terms_φ, quantities, mapping=mapping)
terms_c_fact = dolfiny.units.factorize(terms_c, quantities, mapping=mapping)
```

```
terms_φ_norm = dolfiny.units.normalize(terms_φ_fact, "potential", quantities)
terms_c_norm = dolfiny.units.normalize(terms_φ_fact, "diffusion", quantities)
```

```
form_nondimensional = sum(terms_φ_norm.values()) + sum(terms_c_norm.values())
```

Buckingham Pi analysis

```
==================================================
Buckingham Pi Analysis
==================================================
Symbol | Expression                   | Value (in base units)
-------+----------------------------+------------------------------------------------
c_ref  | 50.0*mole/meter**3           | 50.0*mole/meter**3
φ_ref  | 1.0*volt                     | 1.0*kilogram*meter**2/(ampere*second**3)
D_ref  | 1.0e-10*meter**2/second      | 1.0e-10*meter**2/second
ε_0    | 8.854e-12*farad/meter        | 8.854e-12*ampere**2*second**4/(kilogram*meter**3)
F      | 9.649e+4*coulomb/mole        | 9.649e+4*ampere*second/mole
R      | 8.314*joule/(kelvin*mole)    | 8.314*kilogram*meter**2/(kelvin*mole*second**2)
T      | 300.0*kelvin                 | 300.0*kelvin
l_ref  | angstrom                     | 1.0e-10*meter
e_0    | 1.602e-19*coulomb            | 1.602e-19*ampere*second

Dimension matrix (7 × 9):
Dimension           | c_ref | φ_ref | D_ref | ε_0 | F  | R  | T | l_ref | e_0
--------------------+-------+-------+-------+-----+----+----+---+-------+----
amount_of_substance | 1     | 0     | 0     | 0   | -1 | -1 | 0 | 0     | 0
current             | 0     | -1    | 0     | 2   | 1  | 0  | 0 | 0     | 1
length              | -3    | 2     | 2     | -3  | 0  | 2  | 0 | 1     | 0
luminous_intensity  | 0     | 0     | 0     | 0   | 0  | 0  | 0 | 0     | 0
mass                | 0     | 1     | 0     | -1  | 0  | 1  | 0 | 0     | 0
temperature         | 0     | 0     | 0     | 0   | 0  | -1 | 1 | 0     | 0
time                | 0     | -3    | -1    | 4   | 1  | -2 | 0 | 0     | 1

Dimensionless groups (3):
Group | Expression                                       | Value
------+--------------------------------------------------+-------
Pi_1  | R*T/(F*φ_ref)                                    | 0.0259 thermal voltage
Pi_2  | sqrt(F)*sqrt(c_ref)*l_ref/(sqrt(ε_0)*sqrt(φ_ref)) | 0.0738
Pi_3  | sqrt(F)*sqrt(c_ref)*e_0/(ε_0**(3/2)*φ_ref**(3/2)) | 13.4
==================================================
```

Example | electrodiffusion (Poisson-Nernst-Planck) 4

Form normalisation

```
==================================================
Terms after normalization with "potential"
==================================================
Reference factor from 'potential':
Term      | Factor              | Value (in base units)
----------+---------------------+--------------------------
potential | ε_0*φ_ref**2/l_ref | 0.08854*kilogram/second**2

Term               | Factor                        | Value (in base units)
-------------------+-------------------------------+----------------------
potential          | 1                             | 1.000
electroneutrality  | F*c_ref*l_ref**2/(ε_0*φ_ref) | 0.005449
==================================================
```

```
==================================================
Terms after normalization with "diffusion"
==================================================
Reference factor from 'diffusion':
Term      | Factor                | Value (in base units)
----------+-----------------------+--------------------------------
diffusion | D_ref*c_ref**2/l_ref | 2500.0*mole**2/(meter**5*second)

Term       | Factor                                                | Value (in base units)
-----------+-------------------------------------------------------+----------------------
diffusion  | 1                                                     | 1.000
convection | F*φ_ref/(R*T)                                        | 38.68
debye_0th  | F**2*sqrt(c_ref)*e_0/(R**(3/2)*T**(3/2)*ε_0**(3/2)) | 3213.
debye_1st  | F**3*c_ref*e_0*l_ref/(R**2*T**2*ε_0**2)             | 1475.
==================================================
```

Summary

- Physical units in UFL (user experience, consistency checks)
- Dimensional analysis in UFL (model interpretation, detect sensitivities)
- Non-dimensionalisation of weak forms (improve numerical properties)

- Scaling forms is typically an iterative process

- Scaling is not always (directly) *mathematically* possible
  $\longrightarrow$ expand terms before attempting factorisation
- Scaling is not always (directly) *physically* plausible
  $\longrightarrow$ identification of reference scales
- Scaling is not always (directly) *numerically* beneficial
  $\longrightarrow$ check conditioning, relative participation of terms

`dolfiny.uni.lu`

`fenics-dolfiny/dolfiny`

`pip install dolfiny`