Induction heating verification

Verification of Elmer simulations for induction heating.

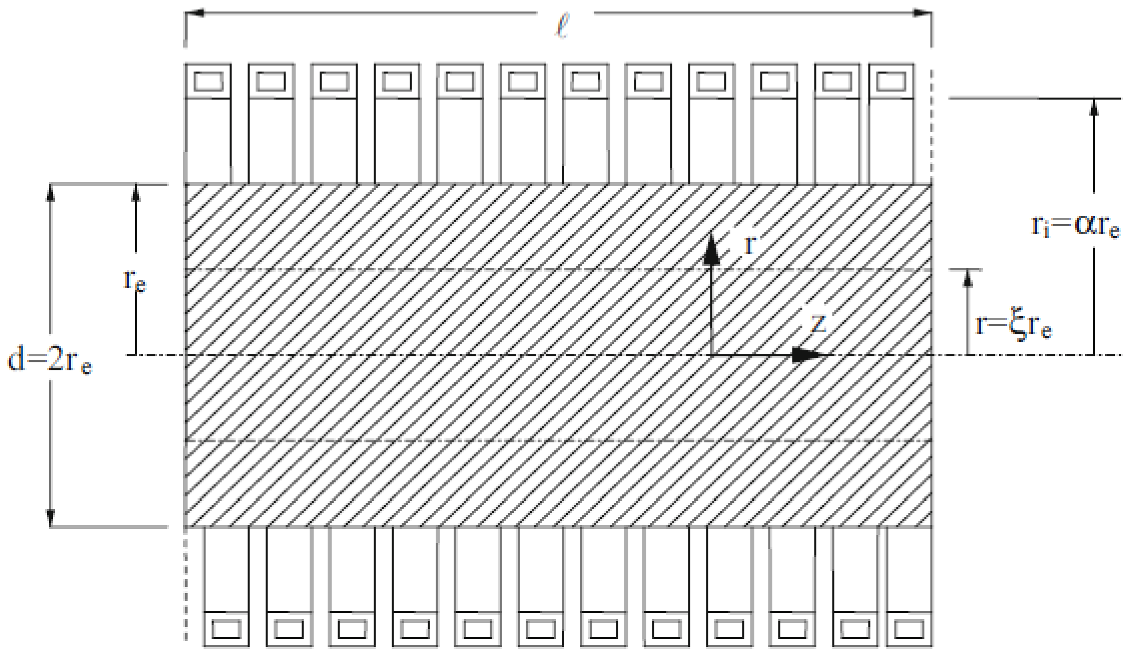
Theory according to S. Lupi: Fundamentals of Electroheat, Chapter 6. Springer, 2017.

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

Sketch

From Lupi2017 p.358:



Assumptions

- Infinite axial length
- Displacement currents neglected
- Sinusoidal waves
- Axisymmetric

Notation

The notation is according to Lupi2017:

- $\dot{H}$ - time dependant value / amplitude and
- H - rms-value

is used for all electromagnetic variables. Under the assumption of sinusoidal waves $H = \frac{\dot{H}}{\sqrt{2}}$ applies.

Cz-Growth parameters

Parameters of induction heating in Cz-Test experiments (Experiments/Test-CZ/Sn growth/2020-06-18 growth_with_ROT_reference to numerics).

Geometry parameters:

$$\begin{aligned} r_e &= 0.035 \text{ m} \\ l_{coil} &= 0.066 \text{m} \\ N_{coil} &= 4.5 \end{aligned}$$

Electromagnetic parameters:

$$\begin{aligned} \dot{I} &= 104.7 \text{ A} \\ f &= 13.4 \cdot 10^3 \text{ Hz} \\ \omega &= 2\pi f \approx 84200 \ \frac{1}{\text{s}} \\ \sigma &= 5.88 \cdot 10^4 \ \frac{\text{S}}{\text{m}} \\ \rho &= \frac{1}{\sigma} \approx 1.7 \cdot 10^{-5} \ \Omega\text{m} \end{aligned}$$

Resulting induction heating parameters:

$$\begin{aligned} \delta &= \sqrt{\frac{2\rho}{\omega\mu_0}} \approx 0.018 \text{ m} \\ m &= \frac{\sqrt{2} r_e}{\delta} \approx 2.75 \\ \dot{H}_e &= \frac{N_{coil} \dot{I}}{l_{coil}} \approx 7138 \ \frac{\text{A}}{\text{m}} \end{aligned}$$

Verification case

Parameters

Selected parameters:

$$\begin{aligned}\delta &= 0.018\\ \dot{H}_e &= 7138\ \frac{\mathrm{A}}{\mathrm{m}}\\ \omega &= 84200\ \frac{1}{\mathrm{s}}\\ r_e &= 0.035\ \mathrm{m}\\ l &= 0.05\ \mathrm{m}\\ N_{coil} &= 4\end{aligned}$$

Resulting parameters:

$$\begin{aligned}m &= \frac{\sqrt{2} r_e}{\delta}\\ \rho &= \frac{\delta^2 \omega \mu_0}{2}\end{aligned}$$

In [2]:

```
delta = 0.018
H = 7138
omega = 84200
r_e = 0.035
l = 0.05

m = 2**0.5 * r_e / delta
print('m =', m)

rho = delta**2 * omega * 4e-7*np.pi / 2
print('rho =', rho, 'Ohm m')
```

```
m = 2.749859704614352
rho = 1.7141032172810484e-05 Ohm m
```

Induction heating

Current density (Lupi2017 p.361 eqn 6.9)

$$\dot{J} = \sqrt{-j} \cdot \dot{H}_e \cdot \frac{\sqrt{2}}{\delta} \cdot \frac{J_1(\sqrt{-j} \cdot m\xi)}{J_0(\sqrt{-j} \cdot m)}$$

with J_0 and J_1 Bessel functions of zero and first order, respectively. An alternative formulation of this equation in Davies1990 p.96 is equal (compared Davies1990 p.96 eqn 9.10 to Lupi2017 p.362 eqn 6.9a). At the outer surface of the cylinder ($\xi = 1$) applies (Lupi2017 p.362 eqn 6.11):

$$\dot{J}_e = \sqrt{-j} \cdot \dot{H}_e \cdot \frac{\sqrt{2}}{\delta} \cdot \frac{J_1(\sqrt{-j} \cdot m)}{J_0(\sqrt{-j} \cdot m)} = -\dot{H}_e \frac{m}{r_e}(P + j \cdot Q)$$

The coefficients P and Q are given by (Lupi2017 p.362 eqn 6.12):

$$P + j \cdot Q = -\sqrt{-j} \cdot \frac{J_1(\sqrt{-j} \cdot m)}{J_0(\sqrt{-j} \cdot m)}$$

In [3]:

```
from scipy.special import jv

P_jQ = -np.sqrt(-1j) * jv(1, np.sqrt(-1j) * m) / jv(0, np.sqrt(-1j) * m)
P = np.real(P_jQ)
Q = np.imag(P_jQ)
print('P = ', P)
```

```
P =  0.5123733794894497
```

The induced power computed from Poynting vector is given by (Lupi2017 p. 367 eqn 6.17):

$$P_w = H_e^2 \frac{\rho}{\delta} \sqrt{2} P 2\pi r_e l = \frac{\dot{H}_e^2}{2} \frac{\rho}{\delta} \sqrt{2} P 2\pi r_e l$$

In [4]:

```
P_w = H**2 / 2 * rho / delta * 2**0.5 * P * 2 * np.pi * r_e * l
print('P_w =', P_w, 'W')
```

```
P_w = 193.2889510284226 W
```

Elmer Simulation

Parameters

For to fulfill the assumption of an infinite domain, the inner section (of length l) of a long coil ($l_{tot} >> 2r_i$) is analyzed. This is controlled by the parameter

$$n_l = \frac{l_{tot}}{l}$$

In [5]:

```
n_l = 10
```

Setup

The following values are set:

$$\begin{aligned} r_e &= 0.035 \text{ m} \\ l &= 0.05 \text{ m} \\ \dot{H}_e &= 7138 \ \frac{\text{A}}{\text{m}} \end{aligned}$$

In [6]:

```
r_e = 0.035
l = 0.05
H = 7138
l_tot = n_l * l
print('l_tot =', l_tot, 'm')
```

```
l_tot = 0.5 m
```

For the coil (modeled by a rectangle) the following values are set:

$$\begin{aligned} N_{coil} &= 1 \\ I &= \frac{\dot{H}_e l_{tot}}{N_{coil}} \\ r_i &= 0.04 \text{ m} \\ d &= 0.01 \text{ m} \end{aligned}$$

In [7]:

```
N = 1
r_i = 0.04
d = 0.01

I = H * l_tot / N
print('I =', I, 'A')
```

```
I = 3569.0 A
```

Geometry modeling

Geometry and mesh generation using Gmsh, using the refinement factor α_{mesh}.

In [8]:

```
alpha_mesh = 1
```

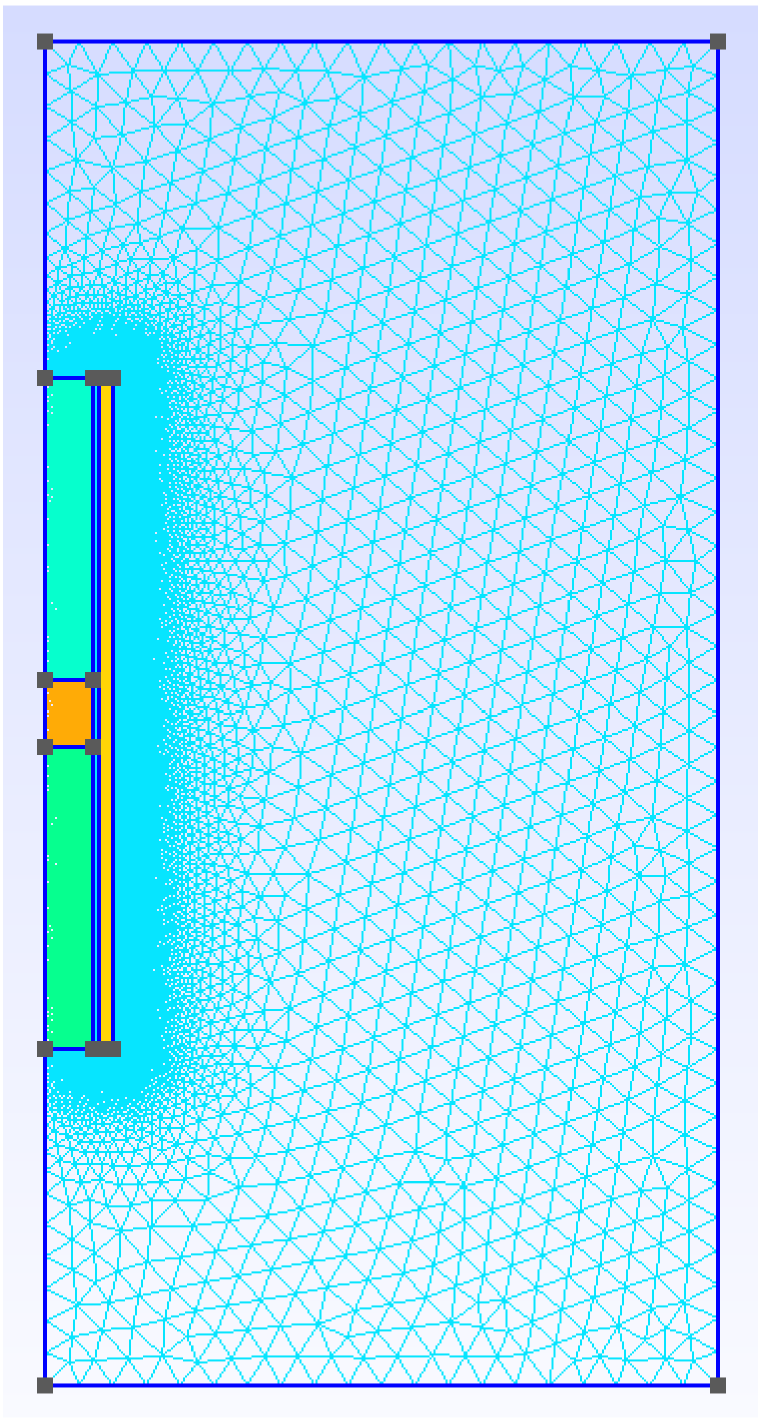
Mesh analyzed by hand

Parameters:

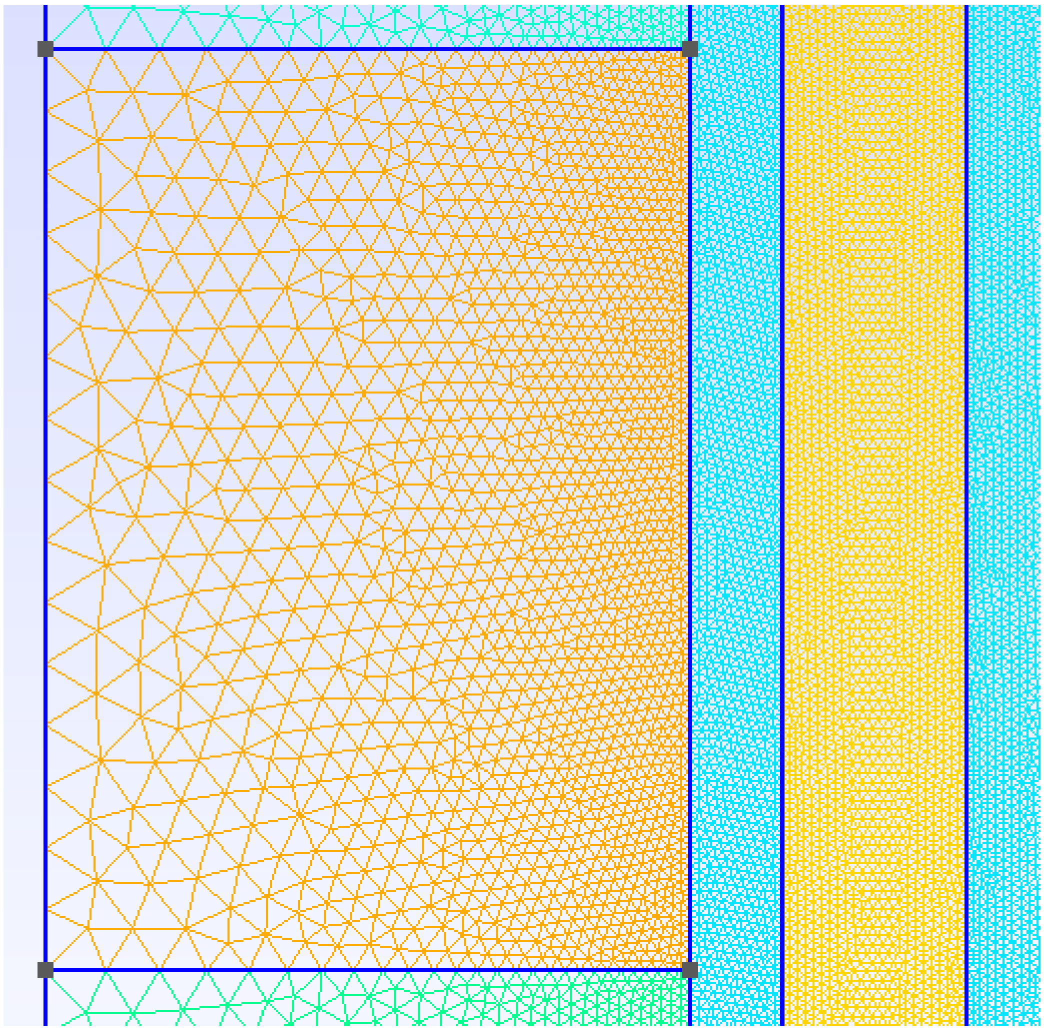
- $n_l = 10$
- α_{mesh} = 1

Number of triangles: 84271

Overview of the mesh:



Zoom at cylinder:



Elmer setup

Solvers

Use solver *MagnetoDynamics2DHarmonic* and *MagnetoDynamicsCalcFields* or *StatMagSolver* to evaluate electromagnetics. These solvers are active in the whole domain.

The solver *HeatSolver* is only active in the cylinder in the middle of the coil, where the induction heating is to be analyzed.

Boundary conditions

Electromagnetics: A potential of zero is prescibed at the outer surfaces.

Thermodynamics: At the top and bottom boundaries (inner cylinder to top and bottom one), no boundary conditions is prescribed, which means that a natural boundary condition with zero heat flux applies. At the outer surface of the cylinder a fixed temperature of 20°C is set.

Evaluation

The heat flux over this boundary, which is in the steady state equal to the induced heat, is analyzed by summing up the *boundary loads* using the *SaveScalars* solver. The value is written to *scalars.dat*. The value needs to be multiplied by 2π to get the total power in Watt

In [9]:

```
from induction_heating import geometry, sif
from pyelmer import execute

ph_cylinder, ph_cylinder_ends, ph_coil, ph_air, ph_cylinder_surf, ph_outside_surfs = ge
ometry(l, n_l, r_e, r_i, d, alpha_mesh=alpha_mesh)
sif(ph_cylinder, ph_cylinder_ends, ph_coil, ph_air, ph_cylinder_surf, ph_outside_surfs,
omega, I, rho, d, l_tot, mgdyn=True)
```

```
Wrote sif-file.
```

In [10]:

```
execute.run_elmer_grid('./simdata/', 'induction_verification.msh2')
execute.run_elmer_solver('./simdata/')
```

Evaluation

Read induced power computed by Elmer and compare it to analytical computed value.

```
In [11]:
```

```python
with open('./simdata/scalars.dat') as f:
    data = f.readlines()
last_iteration = data[-1]
P_w_elmer = -1*float(last_iteration.split('    ')[0]) * 2*np.pi

print('Analytical: \tP_w =', P_w, 'W')
print('Elmer: \t\tP_w =', P_w_elmer, 'W')

dev = (P_w_elmer - P_w) / P_w
print('')
print('Deviation:\t', dev * 100, '%')
```

```
Analytical:     P_w = 193.2889510284226 W
Elmer:          P_w = 189.70607179719764 W

Deviation:       -1.853638923571011 %
```

Parameter study (StatMag solver)

Variable $alpha_{mesh}$:

α_{mesh}	n_l	**Deviation**
4	10	-1.84 %
2	10	-2.37 %
1	10	-2.51 %
0.5	10	-2.54 %

Variable n_l:

α_{mesh}	n_l	**Deviation**
1	10	-2.51 %
1	15	-1.57 %
1	20	-1.28 %
1	25	-1.19 %

Evaluation of joule heat

Analytical solution

$$\dot{J} = \sqrt{-j}\dot{H}_e \frac{\sqrt{2}}{\delta} \frac{J_1(\sqrt{-j} \cdot m\xi)}{J_0(\sqrt{-j} \cdot m)}$$

$$w = \rho J^2 = \rho \frac{\dot{J}^2}{2}$$

Simulation

The base setup using

- $\alpha_{mesh} = 1$ and
- $n_l = 10$

is applied in this comparison.

In [12]:

```
def J(xi, H, delta, m):
    J = (-1j)**0.5 * H * 2**0.5 / delta * jv(1, (-1j)**0.5 * m * xi) / jv(0, (-1j)**0.5
* m)
    return J
```

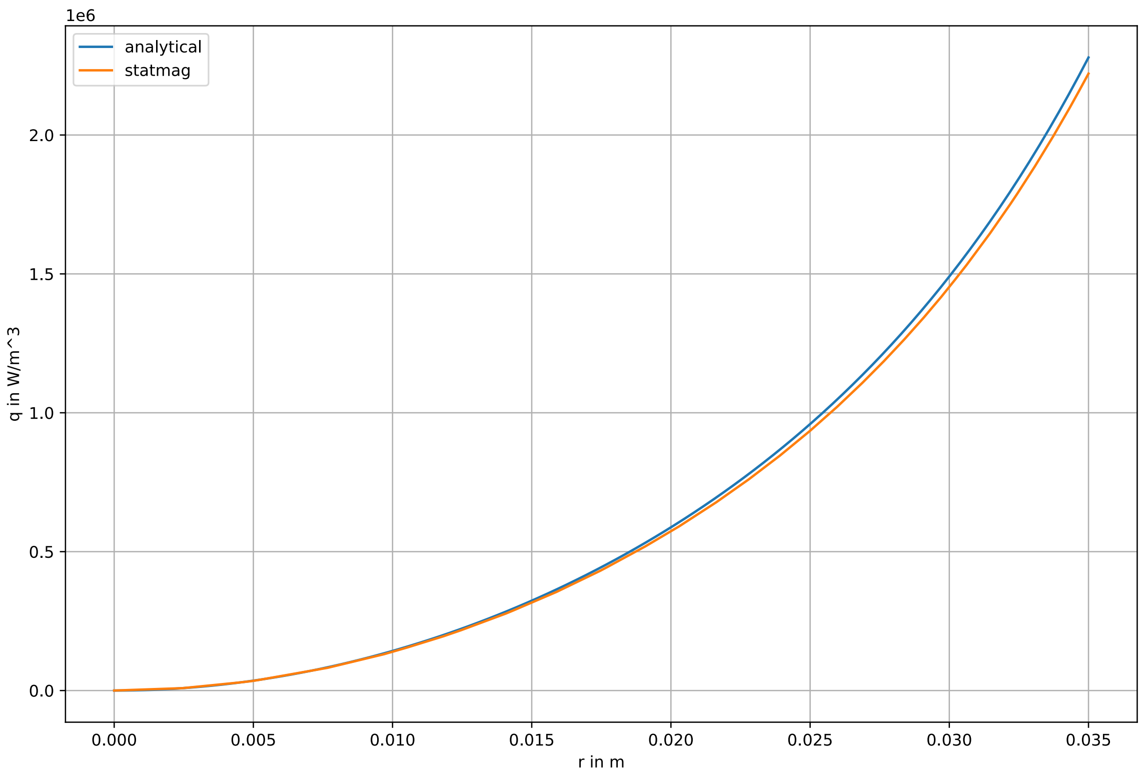
In [13]:

```
# analytical
xi = np.linspace(0, 1, 100)
J_xi = J(xi, H, delta, m)
w = rho * np.abs(J_xi)**2 / 2
# numerical
df = pd.read_csv('./simdata-base/line-data.csv')

fig, ax = plt.subplots(1, 1, figsize=(12,8))
ax.grid()
line, = ax.plot(xi * r_e, w)
line.set_label('analytical')
line, = ax.plot(df['Points:0'], df['joule heating'])
line.set_label('statmag')
ax.legend()
ax.set_xlabel('r in m')
ax.set_ylabel('q in W/m^3')
```

Out[13]:

```
Text(0, 0.5, &#39;q in W/m^3&#39;)
```



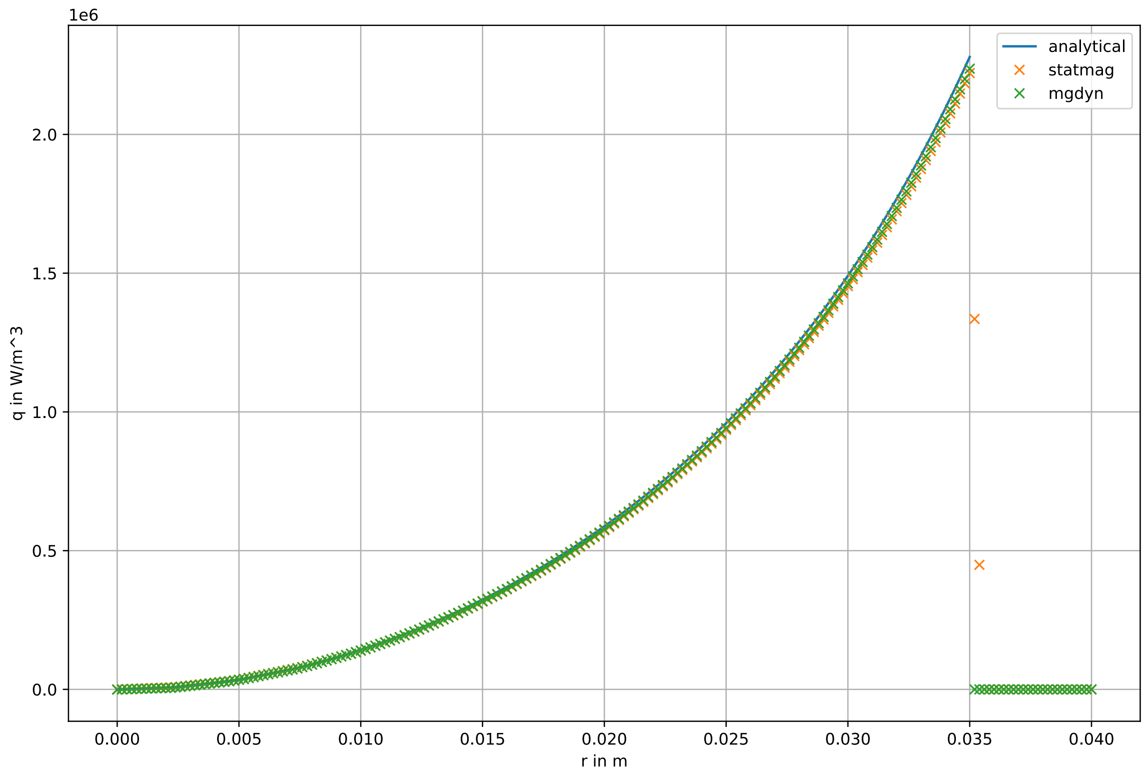
In [14]:

```
# numerical
df_statmag = pd.read_csv('./simdata-statmag-new/line.dat', delim_whitespace=True, heade
r=None)
df_mgdyn = pd.read_csv('./simdata-mgdyn-new/line.dat', delim_whitespace=True, header=No
ne)

fig, ax = plt.subplots(1, 1, figsize=(12,8))
ax.grid()
line, = ax.plot(xi * r_e, w)
line.set_label('analytical')
line, = ax.plot(df_statmag.iloc[:, 3], df_statmag.iloc[:, 11], 'x')
line.set_label('statmag')
line, = ax.plot(df_mgdyn.iloc[:, 3], df_mgdyn.iloc[:, 26], 'x')
line.set_label('mgdyn')
ax.legend()
ax.set_xlabel('r in m')
ax.set_ylabel('q in W/m^3')
```

Out[14]:

```
Text(0, 0.5, &#39;q in W/m^3&#39;)
```


1e6
analytical
statmag
mgdyn
q in W/m^3
r in m