Hyperelastic materials

Kevin M. Moerman[1]

1. *Mechanical Engineering, University of Galway, Galway, Ireland*

2025/05/27

(This is a Pluto notebook featuring the Julia programming language.)

Introduction

- So-called **hyperelastic** formulations are non-linear constitutive (material behaviour) "laws" which are useful to describe nonlinear elastic materials undergoing large (finite strain) deformation.
- In hyperelasticity the constitutive or material law is defined by a so **strain energy density** function often denoted by a W or Ψ symbol.
- The strain energy density Ψ is a **scalar function** (so not a tensor or vector valued function).
- The strain energy density function has units of **energy per unit volume** such as J/m^3.
- However if one recalls that J can be written in terms of Nm, then we see that $J/m^3 = Nm/m^3 = N/m^2$, which means we may equivalently say that Ψ has **units of stress**.

Stress computation

- Derivatives of Ψ with a deformation metric provide a stress metric (there are different types of strains each with their own *work conjugate* stress type).
- For instance, the second *Piola-Kirchoff stress* $\mathbf{S}$ is obtained through the derivative with the *Green-Lagrange strain* $\mathbf{E}$

$$\mathbf{S} = \frac{\partial \Psi}{\partial \mathbf{E}}$$

- We tend to focus on the *true stress* or **Cauchy stress** $\boldsymbol{\sigma}$, which is obtained through with the aid of the *deformation gradient tensor* $\mathbf{F}$:

$$\boldsymbol{\sigma} = J^{-1}\mathbf{F}\mathbf{S}\mathbf{F}^\top$$

- In some cases formulations are specified using the *principal stretches* λ_i. These may also be used to derived (principal) stresses e.g.:

$$\sigma_i = J^{-1}\lambda_i \frac{\partial \Psi}{\partial \lambda_i}$$

Three types of hyperelastic formulations

- Three types of hyperelastic formulation types are treated, here with a focus on the Ogden formulation:

1. **Constrained** formulations (a.k.a "incompressible" formulations)

$$\Psi(\lambda_1, \lambda_2, \lambda_3) = \sum_{a=1}^{N} \frac{c_a}{m_a^2} \left(\lambda_1^{m_a} + \lambda_2^{m_a} + \lambda_3^{m_a} - 3\right)$$

2. **Unconstrained** or **coupled** formulations (a.k.a "compressible" formulations)

$$\Psi(\lambda_1, \lambda_2, \lambda_3) = \frac{\kappa'}{2}(J-1)^2 + \sum_{a=1}^{N} \frac{c_a}{m_a^2} \left(\lambda_1^{m_a} + \lambda_2^{m_a} + \lambda_3^{m_a} - 3 - m_a \ln(J)\right)$$

3. **Uncoupled** formulations (a.k.a "nearly incompressible" formulations)

$$\Psi(\tilde{\lambda}_1, \tilde{\lambda}_2, \tilde{\lambda}_3) = \frac{\kappa}{2} \ln(J)^2 + \sum_{a=1}^{N} \frac{c_a}{m_a^2} \left(\tilde{\lambda}_1^{m_a} + \tilde{\lambda}_2^{m_a} + \tilde{\lambda}_3^{m_a} - 3\right)$$

- This notebook discusses these and provides example implementations for uniaxial loading and first order $N = 1$ **Ogden hyperelastic** formulations.

For more background information see chapter 6 Hyperelasticity" in Holzapfel's book: *G. Holzapfel, Nonlinear solid mechanics: A continuum approach for engineering. John Wiley & Sons Ltd., 2000.*

Side note

It is important to note that the exact implementation of material formulations might depend on the software used. The formulations presented above are implemented in line with their implementation in the finite element solver FEBio (see also here). The exact implementation for a particular finite element implementation might differ, for instance the Abaqus implementation is instead (see also here):

$$\Psi(\tilde{\lambda}_1, \tilde{\lambda}_2, \tilde{\lambda}_3) = \sum_{a=1}^{N} \frac{2\mu_i}{m_a^2} \left(\tilde{\lambda}_1^{m_a} + \tilde{\lambda}_2^{m_a} + \tilde{\lambda}_3^{m_a} - 3\right) + \frac{1}{D_a}(J-1)^{2a}$$

With the initial bulk modulus $\kappa_0 = \frac{2}{D_1}$. Note the difference in terms of $2\mu_a$ instead of c_a, and the use of D factors rather than a single bulk modulus κ. It is important to study the right formulation and to be careful when using parameters from other formulation types seen in the literature. Where possible it is best to fit your model to experimental data.

Anatomy of the Ogden formulation

- Typically the Ogden formulation looks something like this (contrained form showed here, implementations vary depending on software):

$$\Psi(\lambda_1, \lambda_2, \lambda_3) = \sum_{a=1}^{N} \frac{c_a}{m_a^2}\left(\lambda_1^{m_a} + \lambda_2^{m_a} + \lambda_3^{m_a} - 3\right)$$

- If we "distribute" the -3 as a set of -1's, and work a factor $\frac{1}{m_a}$ into the summation we can see the above is equivalent to:

$$\Psi(\lambda_1, \lambda_2, \lambda_3) = \sum_{a=1}^{N} \frac{c_a}{m_a}\left(\frac{1}{m_a}(\lambda_1^{m_a} - 1) + \frac{1}{m_a}(\lambda_2^{m_a} - 1) + \frac{1}{m_a}(\lambda_3^{m_a} - 1)\right)$$

- Now one may recognize these as the Seth-Hill class of strains

$$E_i^{(m_a)} = \frac{1}{m_a}(\lambda_i^{m_a} - 1)$$

- For instance using $m_a = 2$ makes it use the Green-Lagrange strain $\mathbf{E}$

$$E_i = \frac{1}{2}(\lambda_i^2 - 1)$$

- Furthermore we may recognize that the sum of such parts for is actually the trace of such a strain tensor leading to:

$$\Psi = \sum_{a=1}^{N} \frac{c_a}{m_a}\mathrm{tr}(\mathbf{E}^{(m_a)})$$

- So the Ogden formulation is a powerful law where we define energies by scaling (multiplying) the trace of a chosen *"strain type"* (defined by m_a) by shear modulus like parameters c_a. Summing lots of terms ($N > 1$) allows one to capture complex stiffnening behaviour.

- The Ogden formulation can also conveniently be used as the **"mother" of many other formulations**
- Using $N = 1$ and $m_1 = 2$ makes the Ogden formulation reduce to a **Neo-Hookean** formulation

$$\Psi(\lambda_1, \lambda_2, \lambda_3) = \frac{c_1}{4}\left(\lambda_1^2 + \lambda_2^2 + \lambda_3^2 - 3\right) = \frac{c_1}{4}\left(I_1 - 3\right) = \frac{c_1}{2}\mathrm{tr}(\mathbf{E})$$

- The Neo-Hookean is one of the simples hyperelastic formulations and is named after the fact that it can be thought of as an extension of Hooke's law to non-linear solid mechanics (it reduces to Hooke's law for infinitesimal strains).
- Using $N = 2$ and $m_1 = -m_2 = 2$ makes the Ogden formulation reduce to a **Mooney-**

Rivlin formulation (if $J = 1$)

$$\Psi(\lambda_1, \lambda_2, \lambda_3) = \frac{c_1}{4}\left(\lambda_1^2 + \lambda_2^2 + \lambda_3^2 - 3\right) + \frac{c_2}{4}\left(\lambda_1^{-2} + \lambda_2^{-2} + \lambda_3^{-2} - 3\right)$$

$$= \frac{c_1}{4}\left(I_1 - 3\right) + \frac{c_2}{4}\left(I_2 - 3\right)$$

In the above I_1 and I_2 are known as the first and second invariants of the right Cauchy green tensor $\mathbf{C}$. These often appear in the literature.

Defining shared variables used by the example numerical implementations

```
## Loading packages
using GLMakie; GLMakie.activate!() # Used for visualisation
```

```
using Roots # For find_zero based equation solving
```

```
N = 1
```

```
## Define parameters common to all examples

# Define material parameters
N  = 1  # The Ogden law order
```

```
c1 = 1.0
```

```
c1 = 1.0  # The shear modulus like parameter
```

```
m1 = 12.0
```

```
m1 = 12.0  # The non-linearity parameter
```

```
kp = 1000.0
```

```
kp = 1000.0  # Bulk modulus like parameter (used for constrained model)
```

```
k = 1000.0
```

```
k  = kp  # Bulk modulus (used for uncoupled model)

#Derive applied stretch
```

```
appliedStretch = 1.3
```

```
appliedStretch = 1.3  # Set applied stretch
```

```
nDataPoints = 50
```

```
nDataPoints = 50  # Number of data points to use for evaluation and graph
```

```
λ₃ = 1.0:0.006122448979591836:1.3
```

```
λ₃ = range(1.0, appliedStretch, nDataPoints)  # The 3 direction stretch
```

Constrained formulations

- The word "constrained" relates to the fact that incompressible behaviour (no volume change) is enforced in the formulation.
- These formulations are not really used in FEA and instead serve as means to easily derive analytical solutions for incompressible behaviour "by hand".

The constrained Ogden formulation

- The constrained Ogden formulation is often presented as:

$$\Psi(\lambda_1,\lambda_2,\lambda_3)=\sum_{a=1}^{N}\frac{c_a}{m_a^2}\left(\lambda_1^{m_a}+\lambda_2^{m_a}+\lambda_3^{m_a}-3\right)$$

- However, something is missing in the above, namely the treatment of the hydrostatic presssure p and its contribution.

$$\Psi(\lambda_1,\lambda_2,\lambda_3,p)=U(p)+\sum_{a=1}^{N}\frac{c_a}{m_a^2}\left(\lambda_1^{m_a}+\lambda_2^{m_a}+\lambda_3^{m_a}-3\right)$$

- For these constrained forms the contribution $U(p)$ is not derived from the constitutive equation but is instead determined using the boundary conditions.

- Below an example for uniaxial loading is presented.
- The uniaxial load (e.g. a tensile or compressive stretch) is here specified in the 3rd (or Z) direction, which means $\lambda_3\neq 1$.
- Using the "incompressiblity" and uniaxial loading assumption we can formulate some useful relations to help solve for the stress.
- First of all, the uniaxial conditions mean the other "lateral" stretches are equivalent:

$$\lambda_1=\lambda_2$$

- Secondly, if the material is truely incompressible we have $J=\lambda_1\lambda_2\lambda_3=1$, and since $\lambda_1=\lambda_2$ we can derive:

$$J=\lambda_1\lambda_1\lambda_3=\lambda_1^2\lambda_3=1\rightarrow\lambda_1=\lambda_2=\sqrt{\frac{1}{\lambda_3}}=\lambda_3^{-\frac{1}{2}}$$

- Thirdly for uniaxial conditions there is only one none-zero stress, the applied stress $\sigma_3=\sigma_{33}$, therefore:

$$\sigma_1=\sigma_2=\sigma_{11}=\sigma_{22}=0$$

- So now with an assummed $J=1$, the ability to express all stretches in terms of λ_3 (the known applied stretch), and the fact that $\sigma_1=\sigma_2=0$, we are ready to start tackling the full stress evaluation.

- First the Cauchy stress tensor $\boldsymbol{\sigma}$ is defined as:

$$\boldsymbol{\sigma} = \bar{\boldsymbol{\sigma}} - \bar{p}\mathbf{I}$$

- The contribution $\bar{\boldsymbol{\sigma}}$ is derived from the constitutive equation:

$$\Psi(\lambda_1, \lambda_2, \lambda_3) = \sum_{a=1}^{N} \frac{c_a}{m_a^2} \left(\lambda_1^{m_a} + \lambda_2^{m_a} + \lambda_3^{m_a} - 3\right)$$

and is obtained from: $\bar{\sigma}_i = \lambda_i \frac{\partial \Psi}{\partial \lambda_i}$

- Leading to:

$$\bar{\sigma}_i = \sum_{a=1}^{N} \frac{c_a}{m_a} \lambda_i^{m_a}$$

- The next step is to determine $\bar{p}$ in this relation:

$$\boldsymbol{\sigma} = \bar{\boldsymbol{\sigma}} - \bar{p}\mathbf{I}$$

- First lets rewrite the above in terms of the principal components σ_i

$$\sigma_i = -\bar{p} + \sum_{a=1}^{N} \frac{c_a}{m_a} \lambda_i^{m_a}$$

- Next we use $\sigma_1 = \sigma_2 = 0$ to derive an expression for $\bar{p}$:

$$\sigma_1 = \sigma_2 = -\bar{p} + \sum_{a=1}^{N} \frac{c_a}{m_a} \lambda_1^{m_a} = 0$$

$$\rightarrow \bar{p} = \sum_{a=1}^{N} \frac{c_a}{m_a} \lambda_1^{m_a}$$

- Finaly, implementing $\lambda_1 = \lambda_2 = \lambda_3^{-\frac{1}{2}}$ leads to:

$$\rightarrow \bar{p} = \sum_{a=1}^{N} \frac{c_a}{m_a} (\lambda_3^{-\frac{1}{2}})^{m_a} = \sum_{a=1}^{N} \frac{c_a}{m_a} \lambda_3^{-\frac{m_a}{2}}$$

* Which therefore allows for the formulation of an expression for $\bar{\sigma}_3$:

$$\sigma_3 = -\bar{p} + \sum_{a=1}^{N} \frac{c_a}{m_a} \lambda_3^{m_a} = -\sum_{a=1}^{N} \frac{c_a}{m_a} \lambda_3^{-\frac{m_a}{2}} + \sum_{a=1}^{N} \frac{c_a}{m_a} \lambda_3^{m_a}$$

- Which can be simplified to:

$$\sigma_3 = \sum_{a=1}^{N} \frac{c_a}{m_a}\left(\lambda_3^{m_a} - \lambda_3^{-\frac{m_a}{2}}\right)$$

- The full Cauchy stress tensor can then be written as:

$$\boldsymbol{\sigma} = \sigma_3 \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \sum_{a=1}^{N} \frac{c_a}{m_a}\left(\lambda_3^{m_a} - \lambda_3^{-\frac{m_a}{2}}\right) \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- Note that although $\bar{p}$ is a type of pressure contribution, it should not be confused with the full hydrostatic pressure p which is derived from:

$$p = -\frac{1}{3}\mathrm{tr}(\boldsymbol{\sigma}) = -\frac{\sigma_3}{3}$$

Numerical implementation

Compute stresses

```
## The constrained formulation
```

σ₃_con =
[0.0, 0.00932956, 0.0189726, 0.0289623, 0.0393334, 0.0501217, 0.0613645, 0.0731006, 0.085

```
σ₃_con = (c1/m1) .* (λ₃ .^ m1 .- λ₃ .^(-m1/2.0)) # Direct stress computation
```

λ₁_con =
[1.0, 0.996953, 0.993933, 0.990941, 0.987976, 0.985037, 0.982124, 0.979236, 0.976375, 0.9

```
λ₁_con = .√(1.0 ./ λ₃)
```

λ₂_con =
[1.0, 0.996953, 0.993933, 0.990941, 0.987976, 0.985037, 0.982124, 0.979236, 0.976375, 0.9

```
λ₂_con = λ₁_con
```

J_con =
[1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1

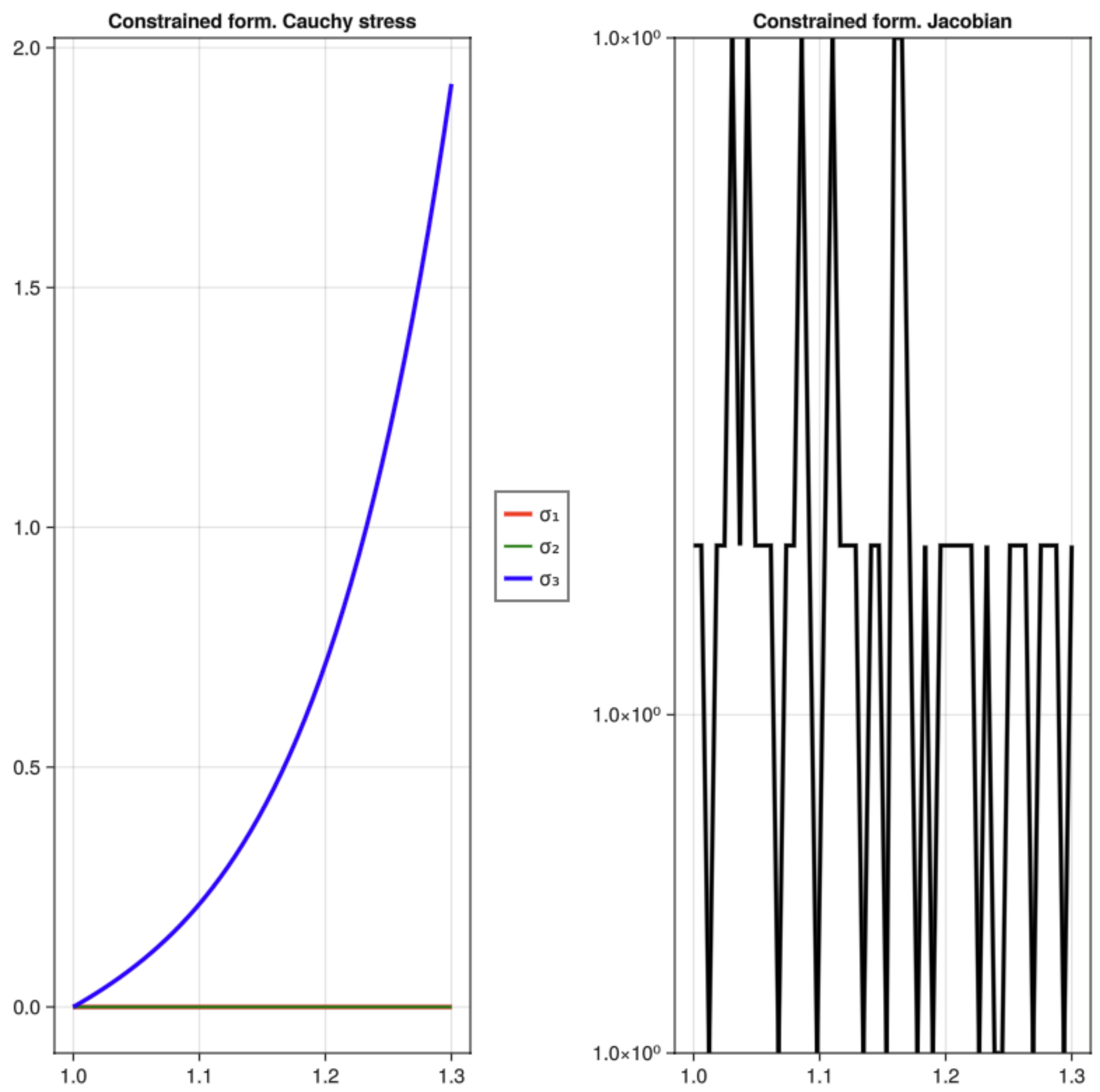
```
J_con = λ₁_con .* λ₂_con .* λ₃
```

σ₁_con =
[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0

```
σ₁_con = zeros(nDataPoints)
```

```
σ₂_con =
 [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, (
```

```julia
σ₂_con = zeros(nDataPoints)
```




```julia
let
    # Visualise stress graphs
    fig = Figure(size = (800, 800))
    ax1 = Axis(fig[1, 1], title="Constrained form. Cauchy stress")
    hs1 = lines!(ax1, λ₃, σ₁_con, color=:red, linewidth=3)
    hs2 = lines!(ax1, λ₃, σ₂_con, color=:green, linewidth=2)
    hs3 = lines!(ax1, λ₃, σ₃_con, color=:blue, linewidth=3)
    Legend(fig[1, 2],[hs1, hs2, hs3],["σ₁", "σ₂", "σ₃"])
    ax2 = Axis(fig[1, 3], title="Constrained form. Jacobian")
    hs4 = lines!(ax2, λ₃, J_con, color=:black, linewidth=3)
    fig
end
```

```
Enter cell code...
```

Unconstrained formulations

These formulations are also known as *coupled* formulations (some literature refers to these formulations as "*compressible*").

- The unconstrained Ogden formulation is given by

$$\Psi(\lambda_1, \lambda_2, \lambda_3) = \frac{\kappa'}{2}(J-1)^2 + \sum_{a=1}^{N} \frac{c_a}{m_a^2}\big(\lambda_1^{m_a} + \lambda_2^{m_a} + \lambda_3^{m_a} - 3 - m_a \ln(J)\big)$$

- The principal Cauchy stresses σ_i can be computed from:

$$\sigma_i = J^{-1}\lambda_i \frac{\partial \Psi}{\partial \lambda_i}$$

- Leading to:

$$\sigma_i = \kappa'\big(J-1\big) + J^{-1}\sum_{a=1}^{N}\frac{c_a}{m_a}\big(\lambda_i^{m_a} - 1\big)$$

Step-by-step derivation:

1. First compute

$$J^{-1}\lambda_i \frac{\partial}{\partial \lambda_i}\big(\frac{\kappa'}{2}(J-1)^2\big)$$

2. Use $J = \lambda_1\lambda_2\lambda_3$, take derivative with respect to λ_3, and use equivalence of result later for the other directions

$$= J^{-1}\lambda_3 \frac{\partial}{\partial \lambda_3}\big(\frac{\kappa'}{2}(\lambda_1\lambda_2\lambda_3 - 1)^2\big)$$

3. Expand square

$$= J^{-1}\lambda_3 \frac{\partial}{\partial \lambda_3}\big(\frac{\kappa'}{2}(\lambda_1^2\lambda_2^2\lambda_3^2 - 2\lambda_1\lambda_2\lambda_3 + 1)\big)$$

9. Now compute the next part:

$$J^{-1}\lambda_i \frac{\partial}{\partial \lambda_i}\left(\sum_{a=1}^{N}\frac{c_a}{m_a^2}\big(\lambda_1^{m_a} + \lambda_2^{m_a} + \lambda_3^{m_a} - 3 - m_a \ln(J)\big)\right)$$

10. First notice that summation can be moved:

$$= \sum_{a=1}^{N} J^{-1} \lambda_i \frac{\partial}{\partial \lambda_i} \left(\frac{c_a}{m_a^2} \left(\lambda_1^{m_a} + \lambda_2^{m_a} + \lambda_3^{m_a} - 3 - m_a \ln(J) \right) \right)$$

11. Next take derivative with respect to λ_3 and aim to use symmetry with respect to any stretch

$$= \sum_{a=1}^{N} J^{-1} \lambda_3 \frac{\partial}{\partial \lambda_3} \left(\frac{c_a}{m_a^2} \left(\lambda_1^{m_a} + \lambda_2^{m_a} + \lambda_3^{m_a} - 3 - m_a \ln(J) \right) \right)$$

12. Use $\frac{\partial}{\partial \lambda_i} \left(\lambda_i^{m_a} \right) = m_a \lambda_i^{m_a - 1}$ and $\ln(J) = \ln(\lambda_1 \lambda_2 \lambda_3) = \ln(\lambda_1) + \ln(\lambda_2) + \ln(\lambda_3)$

$$= \sum_{a=1}^{N} J^{-1} \lambda_3 \frac{c_a}{m_a^2} \left(m_a \lambda_3^{m_a - 1} - \frac{m_a}{\lambda_3} \right)$$

13. Multiply by λ_3 and move J^{-1}

$$= J^{-1} \sum_{a=1}^{N} \frac{c_a}{m_a^2} \left(m_a \lambda_3^{m_a} - m_a \right)$$

14. Simplify by removing m_a factor

$$= J^{-1} \sum_{a=1}^{N} \frac{c_a}{m_a} \left(\lambda_3^{m_a} - 1 \right)$$

15. Generalise for any λ_i:

$$= J^{-1} \sum_{a=1}^{N} \frac{c_a}{m_a} \left(\lambda_i^{m_a} - 1 \right)$$

16. Combine step 8 and 15 to produce overall result:

$$\sigma_i = \kappa' \left(J - 1 \right) + J^{-1} \sum_{a=1}^{N} \frac{c_a}{m_a} \left(\lambda_i^{m_a} - 1 \right)$$

How to compute stresses?

- The stress equations have the unknown J as well as λ_1 and λ_2:

$$\sigma_i = \kappa' \left(J - 1 \right) + J^{-1} \sum_{a=1}^{N} \frac{c_a}{m_a} \left(\lambda_i^{m_a} - 1 \right)$$

- The uniaxial loading conditions and boundary conditions help simplify this to a single unknown
- First of all uniaxial loading in the 3rd or Z-direction means $\lambda_1 = \lambda_2$

- Next we can use the definition of the Jacobian to come to expressions for λ_1 and λ_2
- Since we have $J=\lambda_1\lambda_2\lambda_3$, and $\lambda_1=\lambda_2$ we can derive:

$$J=\lambda_1\lambda_2\lambda_3=\lambda_1\lambda_1\lambda_3=\lambda_1^2\lambda_3$$

$$\rightarrow \lambda_1=\lambda_2=\sqrt{\frac{J}{\lambda_3}}$$

- The above shows that although λ_3 is known, knowledge of J is required in order to determine λ_1 and λ_2. Or conversely λ_1 (or λ_2) needs to be determined allowing for the computation of J. Eitherway one unknown remains.
- To solve for the unkown J we may use the fact that $\sigma_1=\sigma_2=0$

$$\sigma_1=\kappa'\big(J-1\big)+J^{-1}\sum_{a=1}^{N}\frac{c_a}{m_a}\big(\lambda_1^{m_a}-1\big)=0$$

- If we solve for J we can use $\lambda_1=\sqrt{\frac{J}{\lambda_3}}$ and write:

$$\sigma_1=\kappa'\big(J-1\big)+J^{-1}\sum_{a=1}^{N}\frac{c_a}{m_a}\left(\left(\frac{J}{\lambda_3}\right)^{\frac{m_a}{2}}-1\right)$$

- Or if instead we solve for λ_1 we can use $J=\lambda_1^2\lambda_3$ and write:

$$\sigma_1=\kappa'\big((\lambda_1^2\lambda_3)-1\big)+\frac{1}{\lambda_1^2\lambda_3}\sum_{a=1}^{N}\frac{c_a}{m_a}\big(\lambda_1^{m_a}-1\big)=0$$

- Solving these is not trivial but numerical solutions are derived below for J

Numerical implementation

Compute stresses

```
J_uncon =
  [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, (
```

```julia
## The unconstrained or coupled formulation

# One approach is to define a function for S1 and to find the J for which it is zero.
# For this application the find_zero function from the Roots package is useful to find J for σ₁(J)=0.

#Compute Jacobian given boundary conditions σ₁=σ₂=0.0
J_uncon=zeros(nDataPoints)  #Initialize an arry of J values which are all zeros
```

```
for q=1:1:nDataPoints #Loop over all data points
    #Create stress function with current lambda
    σ₁(J) = kp*(J-1.0)+(1.0/J)*(c1/m1)*((sqrt(J/λ₃[q]).^m1)-1.0)

    #Find Jacobian for zero stress, use J=1 as initial
    J_uncon[q]=find_zero(σ₁,1.0)  #Find root of nonlinear function
end
```

```
[1.0, 1.0, 1.00001, 1.00001, 1.00001, 1.00001, 1.00002, 1.00002, 1.00002, 1.00002, 1.0000
```

```
J_uncon
```

```
λ₁_uncon =
 [1.0, 0.996954, 0.993936, 0.990945, 0.987981, 0.985043, 0.982132, 0.979246, 0.976385, 0.9
```

```
#Compute transverse stretches using J values
λ₁_uncon = sqrt.(J_uncon./λ₃)
```

```
λ₂_uncon =
 [1.0, 0.996954, 0.993936, 0.990945, 0.987981, 0.985043, 0.982132, 0.979246, 0.976385, 0.9
```

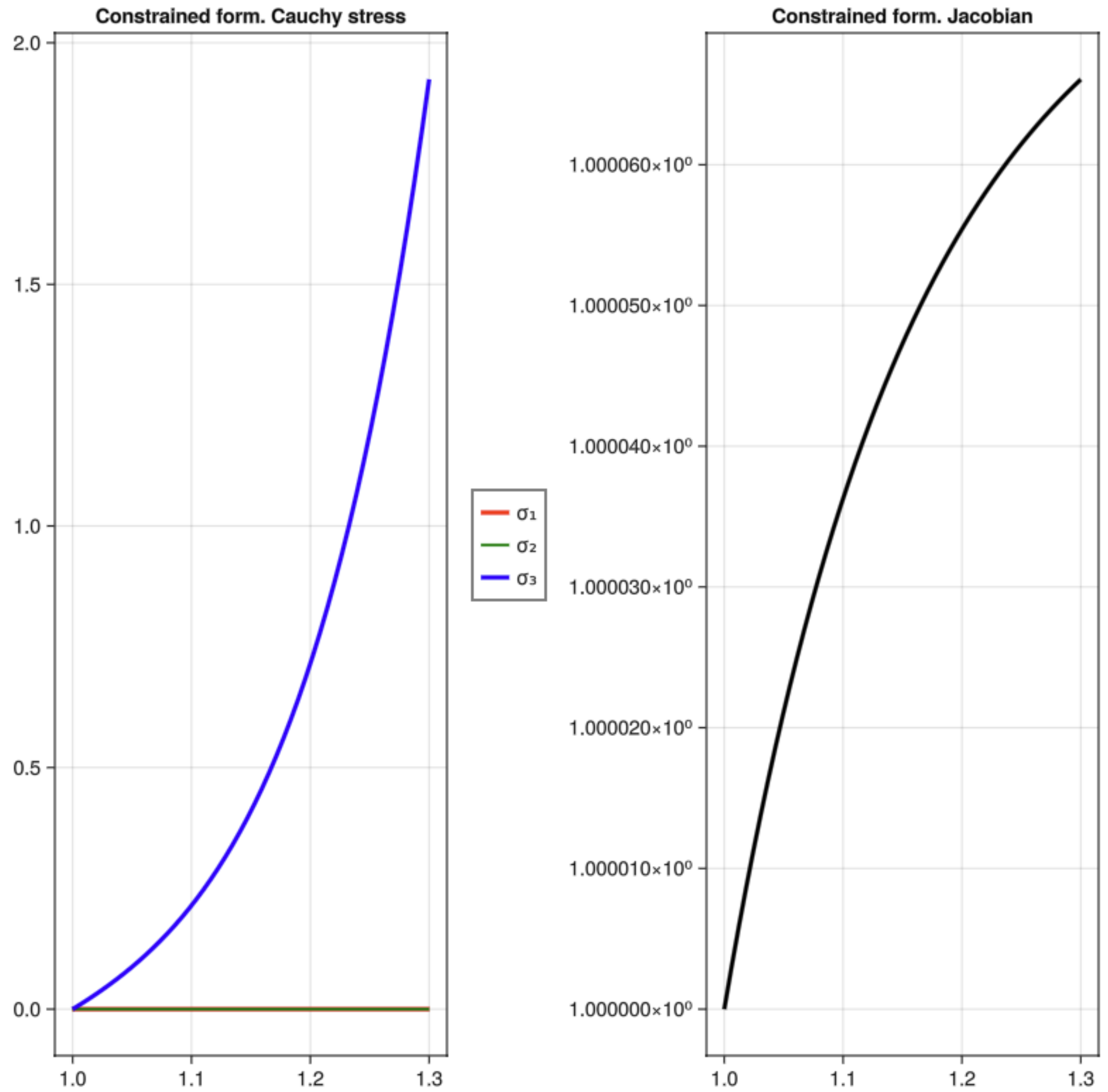
```
λ₂_uncon = λ₁_uncon  #Due to uniaxial loading
```

```
unconstrainedStress (generic function with 1 method)
```

```
#Set up function to ompute principal stresses
function unconstrainedStress(c1, m1, kp, J, λ₁, λ₂, λ₃)
    σ(λ) = kp .* (J .- 1.0) .+ (1.0 ./ J) .* (c1/m1) .* ((λ.^m1) .- 1.0)
    return σ(λ₁), σ(λ₂), σ(λ₃)
end
```

```
([0.0, 8.21804e-14, 1.03181e-13, 2.89699e-16, -1.55605e-15, -9.72642e-14, 4.32536e-14, 6.
```

```
σ₁_uncon, σ₂_uncon, σ₃_uncon = unconstrainedStress(c1, m1, kp, J_uncon, λ₁_uncon,
λ₂_uncon, λ₃)
```




```
let
    # Visualise stress graphs
    fig = Figure(size = (800, 800))
    ax1 = Axis(fig[1, 1], title="Constrained form. Cauchy stress")
    hs1 = lines!(ax1, λ₃, σ₁_uncon, color=:red, linewidth=3)
    hs2 = lines!(ax1, λ₃, σ₂_uncon, color=:green, linewidth=2)
    hs3 = lines!(ax1, λ₃, σ₃_uncon, color=:blue, linewidth=3)
    Legend(fig[1, 2],[hs1, hs2, hs3],["σ₁", "σ₂", "σ₃"])
    ax2 = Axis(fig[1, 3], title="Constrained form. Jacobian")
    hs4 = lines!(ax2, λ₃, J_uncon, color=:black, linewidth=3)
    fig
end
```

Uncoupled formulations

- Given the numerical difficulties in handling truely incompressible behaviour (theoretically requiring $\kappa = \infty$) a special class of constitutive formulations has been developed referred to as *uncoupled* formulations.

- These uncoupled formulations are useful to model nearly-incompressible behaviour
- The term *uncoupled* relates to the fact that strain energy density Ψ is split into two additively seperated parts, namely:

1. A purely *deviatoric* (or isochoric = no volume change) part relating to shape change only Ψ_{dev}
2. A purely *volumetric* part relating to volume change only Ψ_{vol}

$$\Psi = \Psi_{dev} + \Psi_{vol}$$

Uncoupling the deformation

- To accomodate the split special shape and volume changing deformation metrics are required.
- The Jacobian or volume ratio J is already suitable to describe volume change ($J = 0.9$ means 10# volume loss, $J = 1.1$ means 10# volume gain).
- From the definition $J = \lambda_1\lambda_2\lambda_3$ one could imagine a single "spherical" average stretch λ which is the same in all directions such that:

$$J = \lambda_1\lambda_2\lambda_3 = \lambda\lambda\lambda = \lambda^3 \rightarrow \lambda = J^{\frac{1}{3}}$$

- To "take away" the effect of this spherical volume changing stretch λ from each of the stretches we can multiply them by $\frac{1}{\lambda} = J^{-\frac{1}{3}}$:

$$\tilde{\lambda}_i = J^{-\frac{1}{3}}\lambda_i$$

- This introduces the *deviatoric stretches* denoted $\tilde{\lambda}_i$
- We can check if these deviatoric stretches really only change the shape by computing $\tilde{J}$ which should be 1 in magnitude for all stretches:

$$\tilde{J} = \tilde{\lambda}_1\tilde{\lambda}_2\tilde{\lambda}_3 = J^{-\frac{1}{3}}\lambda_1 J^{-\frac{1}{3}}\lambda_2 J^{-\frac{1}{3}}\lambda_3 = J^{-\frac{1}{3}}J^{-\frac{1}{3}}J^{-\frac{1}{3}}\lambda_1\lambda_2\lambda_3 = \frac{1}{J}J = 1$$

The uncoupled Ogden formulation

- The uncoupled Ogden formulation is given as:

$$\Psi(\tilde{\lambda}_1, \tilde{\lambda}_2, \tilde{\lambda}_3) = \frac{\kappa}{2}\ln{(J)}^2 + \sum_{a=1}^{N} \frac{c_a}{m_a^2}\left(\tilde{\lambda}_1^{m_a} + \tilde{\lambda}_2^{m_a} + \tilde{\lambda}_3^{m_a} - 3\right)$$

- Where

$$\Psi_{vol} = \frac{\kappa}{2}\ln{(J)}^2$$

and

$$\Psi_{dev} = \sum_{a=1}^{N} \frac{c_a}{m_a^2} \left(\tilde{\lambda}_1^{m_a} + \tilde{\lambda}_2^{m_a} + \tilde{\lambda}_3^{m_a} - 3\right)$$

- The principal Cauchy stresses σ_i can be computed from:

$$\boldsymbol{\sigma} = \boldsymbol{\sigma}_{vol} + \boldsymbol{\sigma}_{dev}$$

- The volumetric stress $\boldsymbol{\sigma}_{vol}$ is derived from:

$$\boldsymbol{\sigma}_{vol} = p\mathbf{I}$$

where the hydrostatic pressure is now derived directly from the constitutive equation:

$$p = \frac{\partial \Psi_{vol}}{\partial J}$$

resulting in:

$$\boldsymbol{\sigma}_{vol} = \kappa \frac{\ln(J)}{J} \mathbf{I}$$

- The deviatoric stress $\boldsymbol{\sigma}_{dev}$ is derived from:

$$\sigma_{dev_i} = J^{-1} \lambda_i \frac{\partial \Psi_{dev}}{\partial \lambda_i} = J^{-1} \left(\tilde{\lambda}_i \frac{\partial \Psi_{dev}}{\partial \tilde{\lambda}_i} - \frac{1}{3} \sum_{j=1}^{3} \tilde{\lambda}_j \frac{\partial \Psi_{dev}}{\partial \tilde{\lambda}_j} \right)$$

- Since $J = \lambda_1 \lambda_2 \lambda_3$, and $\lambda_1 = \lambda_2$ (due to uniaxial loading in the 3rd direction) we can derive:

$$J = \lambda_1 \lambda_2 \lambda_3 = \lambda_1 \lambda_1 \lambda_3 = \lambda_1^2 \lambda_3$$

$$\rightarrow \lambda_1 = \lambda_2 = \sqrt{\frac{J}{\lambda_3}}$$

- Using

$$\tilde{\lambda}_i \frac{\partial \Psi_{dev}}{\partial \tilde{\lambda}_i} = \sum_{a=1}^{N} \frac{c_a}{m_a} \tilde{\lambda}_i^{m_a}$$

we can formulate

$$\sigma_{dev_i} = J^{-1} \sum_{a=1}^{N} \frac{c_a}{m_a} \left(\tilde{\lambda}_i^{m_a} - \frac{1}{3} \left(\tilde{\lambda}_1^{m_a} + \tilde{\lambda}_2^{m_a} + \tilde{\lambda}_3^{m_a} \right) \right)$$

$$\sigma_{dev_i} = J^{-1}\sum_{a=1}^{N}\frac{c_a}{m_a}\left(\tilde{\lambda}_i^{m_a} - \frac{1}{3}\left(\tilde{\lambda}_1^{m_a} + \tilde{\lambda}_2^{m_a} + \tilde{\lambda}_3^{m_a}\right)\right)$$

- Since $\tilde{\lambda}_1\tilde{\lambda}_2\tilde{\lambda}_3 = 1$ one may use $\tilde{\lambda}_1 = \tilde{\lambda}_2 = \sqrt{\frac{1}{\tilde{\lambda}_3}} = \tilde{\lambda}_3^{-\frac{1}{2}}$, and therefore:

$$\sigma_{dev_i} = J^{-1}\sum_{a=1}^{N}\frac{c_a}{m_a}\left(\tilde{\lambda}_i^{m_a} - \frac{1}{3}\left(2\tilde{\lambda}_3^{-\frac{m_a}{2}} + \tilde{\lambda}_3^{m_a}\right)\right)$$

- Leading to:

$$\sigma_i = \kappa\frac{\ln(J)}{J} + J^{-1}\sum_{a=1}^{N}\frac{c_a}{m_a}\left(\tilde{\lambda}_i^{m_a} - \frac{1}{3}\left(2\tilde{\lambda}_3^{-\frac{m_a}{2}} + \tilde{\lambda}_3^{m_a}\right)\right)$$

- Which using $\tilde{\lambda}_1 = \tilde{\lambda}_2 = \tilde{\lambda}_3^{-\frac{1}{2}}$ gives the following for σ_1 and σ_2:

$$\sigma_1 = \sigma_2 = \kappa\frac{\ln(J)}{J} + J^{-1}\sum_{a=1}^{N}\frac{c_a}{m_a}\left(\tilde{\lambda}_3^{-\frac{m_a}{2}} - \frac{1}{3}\left(2\tilde{\lambda}_3^{-\frac{m_a}{2}} + \tilde{\lambda}_3^{m_a}\right)\right) = 0$$

- Which using $\tilde{\lambda}_3 = J^{-\frac{1}{3}}\lambda_3$ can be expressed in terms of λ_3 as:

$$\sigma_1 = \sigma_2 = 0 =$$

$$\kappa\frac{\ln(J)}{J} + J^{-1}\sum_{a=1}^{N}\frac{c_a}{m_a}\left((J^{-\frac{1}{3}}\lambda_3)^{-\frac{m_a}{2}} - \frac{1}{3}\left(2(J^{-\frac{1}{3}}\lambda_3)^{-\frac{m_a}{2}} + (J^{-\frac{1}{3}}\lambda_3)^{m_a}\right)\right)$$

- Numerical metods are now needed to solve for J such that $\sigma_1 = \sigma_2 = 0$

```
J_uncoup =
 [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, (
```

```julia
## The uncoupled formulation
# One approach is to define a function for σ₁ and to find the J for which it is zero. For this application the find_zero function is useful to find J for σ₁(J)=0.

#Compute Jacobian given boundary conditions S1=S2=0
J_uncoup = zeros(nDataPoints)  #Initialize an arry of J values which are all zeros
```

```julia
for q=1:1:nDataPoints #Loop over all data points
    #Create stress function with current lambda
    σ₁(J) = k*(log(J)/J)+(1.0/J)*(c1/m1)*( (J^(-1.0/3.0)*λ₃[q])^(-m1/2.0)-(1.0/3.0)*(2.0*(J^(-1.0/3.0)*λ₃[q])^(-m1/2.0)+(J^(-1.0/3.0)*λ₃[q])^m1) )

    #Find Jacobian for zero stress, use J=1 as initial
    J_uncoup[q]=find_zero(σ₁,1.0)  #Find root of nonlinear function
end
```

```
λ₁_uncoup =
  [1.0, 0.996954, 0.993936, 0.990946, 0.987982, 0.985045, 0.982134, 0.979248, 0.976389, 0.9
```

```julia
#Compute transverse stretches using J values
λ₁_uncoup = sqrt.(J_uncoup./λ₃)
```

```
λ₂_uncoup =
  [1.0, 0.996954, 0.993936, 0.990946, 0.987982, 0.985045, 0.982134, 0.979248, 0.976389, 0.9
```

```julia
λ₂_uncoup = λ₁_uncoup  #Due to uniaxial loading
```

```
uncoupledStress (generic function with 1 method)
```

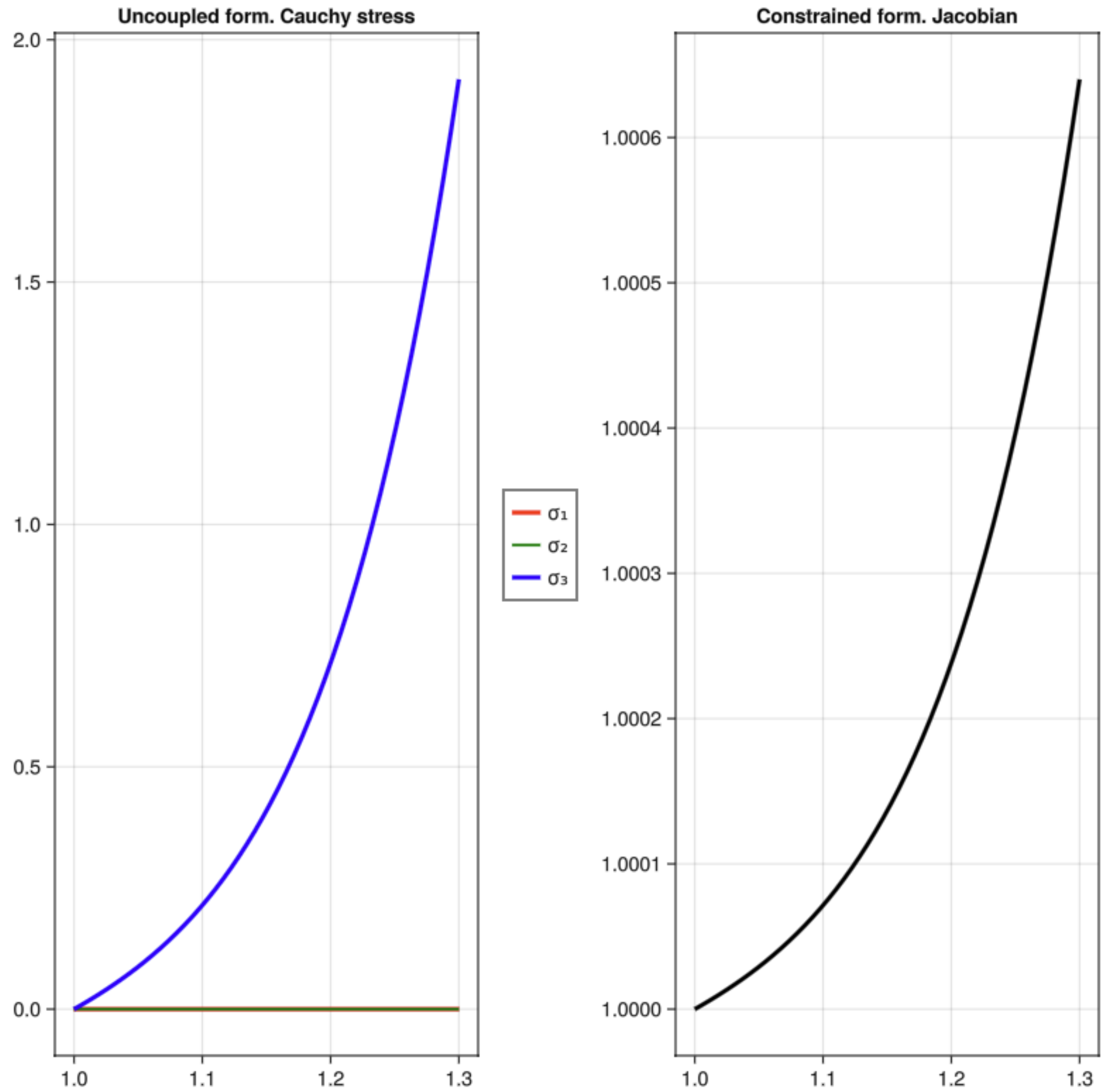
```julia
#Set up function to compute principal stresses
function uncoupledStress(c1, m1, k, J, λ₁, λ₂, λ₃)
    part2 = (1.0/3.0) .* (
          (J .^ (-1.0/3.0) .* λ₁) .^ m1
       .+ (J .^ (-1.0/3.0) .* λ₂) .^ m1
       .+ (J .^ (-1.0/3.0) .* λ₃) .^ m1)

    σ(J,λ) = k .* (log.(J)./J) .+ (1.0 ./ J) .* ((c1/m1) .* ((J.^(-1/3) .* λ).^m1
    .- part2))

    return σ(J,λ₁), σ(J,λ₂), σ(J,λ₃)
end
```

```
([0.0, -9.64337e-14, -2.25289e-14, 7.61995e-14, 6.91339e-14, 7.6366e-14, -1.90473e-15, 1.
```

```julia
σ₁_uncoup, σ₂_uncoup, σ₃_uncoup = uncoupledStress(c1, m1, k, J_uncoup, λ₁_uncoup,
λ₂_uncoup, λ₃)
```




```julia
let
    # Visualise stress graphs
    fig = Figure(size = (800, 800))
    ax1 = Axis(fig[1, 1], title="Uncoupled form. Cauchy stress")
    hs1 = lines!(ax1, λ₃, σ₁_uncoup, color=:red, linewidth=3)
    hs2 = lines!(ax1, λ₃, σ₂_uncoup, color=:green, linewidth=2)
    hs3 = lines!(ax1, λ₃, σ₃_uncoup, color=:blue, linewidth=3)
    Legend(fig[1, 2],[hs1, hs2, hs3],["σ₁", "σ₂", "σ₃"])
    ax2 = Axis(fig[1, 3], title="Constrained form. Jacobian")
    hs4 = lines!(ax2, λ₃, J_uncoup, color=:black, linewidth=3)
    fig
end
```