Table of Contents

Section 3.3 - Stability Analysis
Gibbs tangent-plane condition
Unconstrained formulation
Implementation
1. State and model specification
2. Generate initial guesses
3. Minimise our objective function
Problems and difficulties
Footnotes

Section 3.3 - Stability Analysis

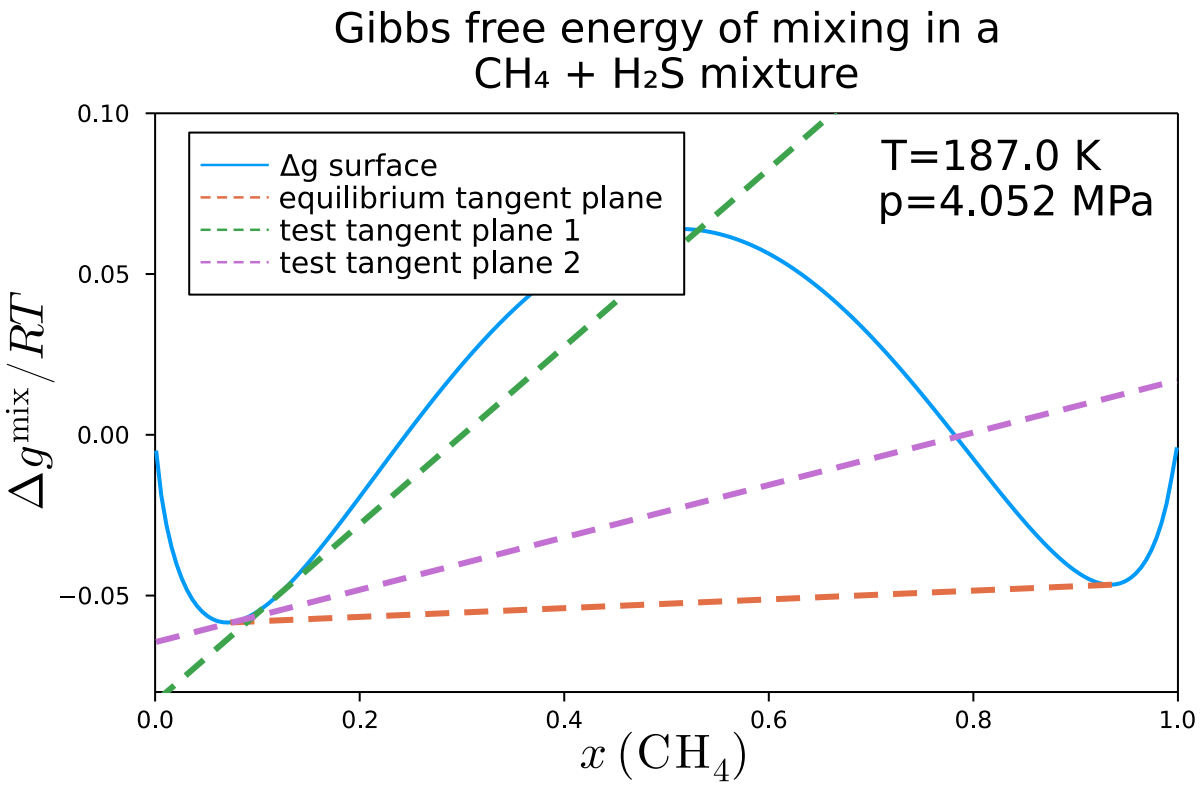
Stability analysis deals with the question "**does a phase split occur?**". This can be very important as it allows us to determine whether further calculation is necessary, or if we can simply evaluate the phase with the lowest Gibbs free energy (G). It is a key part of multiphase flash algorithms, which you will see in section 3.5, as well as providing good initial guesses to our two-phase flash, which we will develop in section 3.4.

Gibbs tangent-plane condition

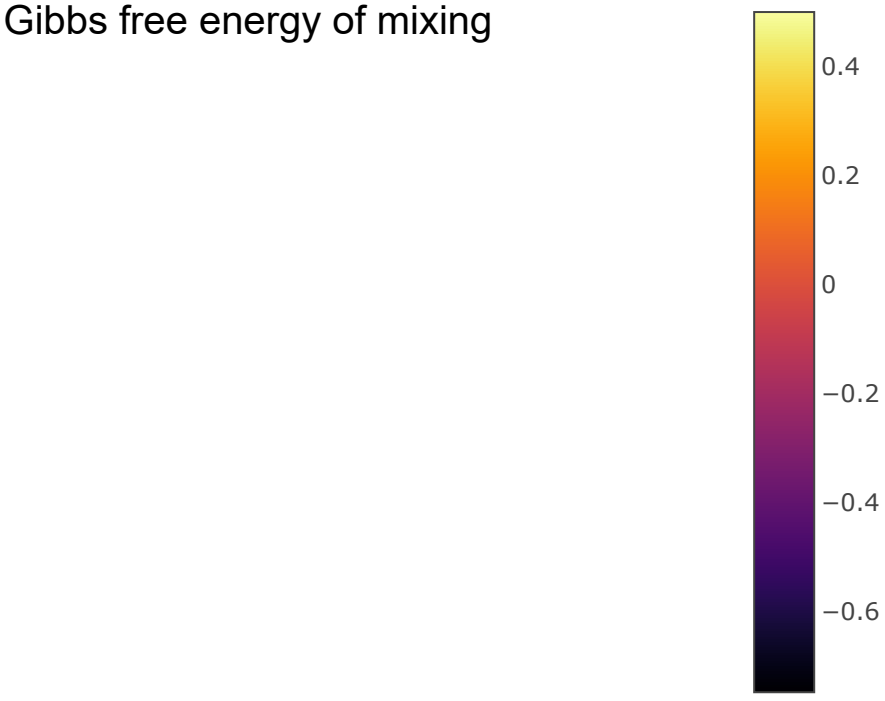
The fundamental description of stability is the Gibbs tangent-plane. We know a system always exists at the minima on the Gibbs free energy surface, so when a tangent plane with a lower G can be constructed, we know that a phase split will occur, as this multiple phase solution will be more stable than the single phase.

For a binary mixture this can be quite easy to visualise. In the example of methane and hydrogen sulfide below you can see the mixture will split into two phases, shown by the intersection of the "equilibrium tangent plane" with the gibbs free energy of mixing surface.

It's important to note that while infinite tangent planes exist, there is only a single **cotangent** plane, representing the equilibrium compositions.



As we move to mixtures with more components - such as the octane, ethane, and propane ternary mixture represented below - it can become a lot harder to tell. For mixtures of many components, this can become impossible to fully visualise.



There is therefore a need for a mathematical description of phase stability that we can apply to any number of components and phases.

This description can be derived by considering the fact the Gibbs free energy being at a **global minimum**. This requires that the formation of any new phase must increase G.

The change in Gibbs free energy upon the formation of δe of a new phase with composition $\mathbf{x}$ from a prexisting mixture with composition $\mathbf{z}$ can be written as

$$\delta G = \delta e \sum x_i(\mu_i(\mathbf{x}) - \mu_i(\mathbf{z}))\ .$$

For a mixture to be stable, we must therefore have that

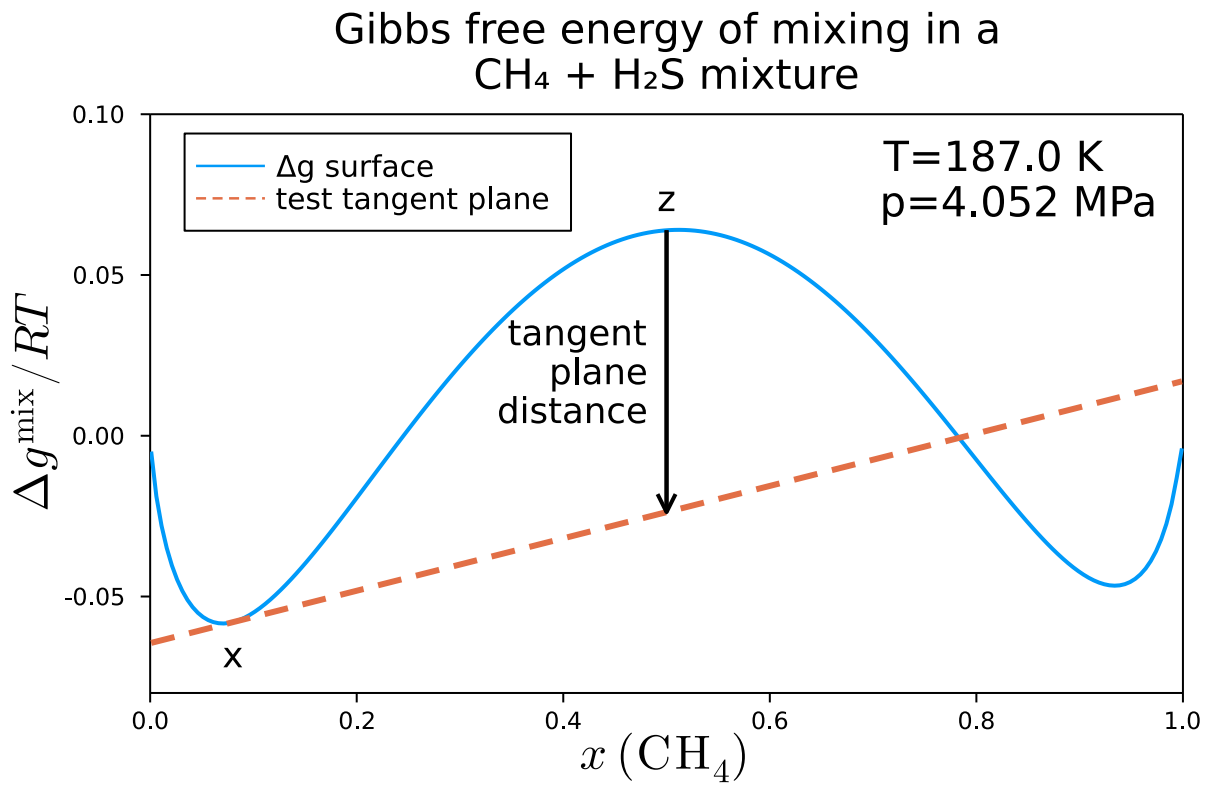
$$\delta G \geq 0 \implies \sum x_i(\mu_i(\mathbf{x}) - \mu_i(\mathbf{z})) \geq 0$$

Physically, this represents the distance from the gibbs free energy surface at overall composition $\mathbf{z}$ to a tangent plane constructed at composition $\mathbf{x}$. Because of this we call this function the **tangent plane distance** function, or

$$TPD(\mathbf{x}) = \sum x_i(\mu_i(\mathbf{x}) - \mu_i(\mathbf{z}))$$

and we can see that if $TPD \geq 0$ for all possible compositions $\mathbf{x}$, then our mixture must be stable. Another way of looking at this is is that if we evaluate every minima of TPD, and they are all positive, then the mixture is stable.

The TPD function can be seen visualised below:



Unconstrained formulation

We now have a function describing whether a mixture is stable or not, but how do we go about implementing this?

If we were to attempt to naively program this in, we would end up with a poorly scaled constrained minimisation problem, requiring all our mole fractions sum to one. While this is possible to solve, it is more complicated than necessary.

$$\min TPD(p^{\text{spec}}, T^{\text{spec}}, \mathbf{z}^{\text{spec}}, \mathbf{x})$$

subject to

$$0 \leq w_i \leq 1 \quad \forall i \in [1, C]$$
$$\sum w_i = 1$$

The first change we make is expressing this in terms of the fugacity coefficients, as typically these are easier to work with. While this isn't technically true when using Clapeyron, it is still a convention that we will follow. To do this, we rewrite our chemical potential in terms of our fugacity coefficients

$$\mu_i = \mu_i^* + RT \ln \frac{f_i}{P^{\text{ref}}} = \mu_i^* + RT(\ln x_i + \ln \varphi_i)$$

noting that P^{ref} is taken as one. Our function is now

$$TPD(\mathbf{x}) = \sum x_i(\ln x_i + \ln \varphi_i(\mathbf{x}) - \ln x_i - \ln \varphi_i(\mathbf{z}))$$

and we then define a "helper variable", d_i, to simplify our expression

$$d_i = \ln z_i - \ln \varphi_i(\mathbf{z})$$

To better scale our problem we use the same RT scaling factor as before

$$tpd = \frac{TPD}{RT} = \sum x_i(\ln x_i + \ln \varphi_i(\mathbf{x}) - d_i)$$

where

$$d_i = \ln z_i + \ln \varphi_i(\mathbf{z})$$

Finally, we remove the mass balance constraints by reformulating changing it to be a function of mole numbers, $\mathbf{X}$,

$$tm(\mathbf{X}) = 1 + \sum_i X_i(\ln X_i + \ln \varphi_i(\mathbf{X}) - d_i - 1)\ .$$

Because of this change tm no longer describes the tangent plane distance, however it can be shown that [1]

$$\begin{aligned} tm < 0 &\iff TPD < 0 \\ \min tm &\iff \min TPD \end{aligned}$$

meaning tm can be used nearly identically to TPD in the context of stability analysis. We should also note that a positive value of tm, does **not necessarily** imply a positive value of TPD. This means that if no negative value of tm is located, we should also check the value of TPD at each minima **[2]**.

$$\min tm(p^{\text{spec}}, T^{\text{spec}}, \mathbf{z}^{\text{spec}}, \mathbf{X})$$

subject to

$$0 \leq X_i \quad \forall i \in [1, C]$$

To satisfy the the final constraint, we use logs. To do this we change our iteration variable from $\mathbf{X}$ to $\log \mathbf{X}$

$$\min tm(p^{\text{spec}}, T^{\text{spec}}, \mathbf{z}^{\text{spec}}, 10^{\log \mathbf{X}})$$

This method is applicable to any number of components, though is easiest to visualise for a binary mixture.

Implementation

Now we're familiar with our problem, and we have an objective function let's implement it.

This will be split into three stages:

1. State and model specification:

- Specify $p, T, \mathbf{z}$

2. Generate initial guesses

- Use a correlation to obtain $\mathbf{x}_0$

3. Minimise our objective function tm

- Use an optimisation algorithm to minimise the tangent plane distance

1. State and model specification

We're going to use a predictive cubic, EPPR78, to capture nonidealities with the binary interaction coefficient. The components, temperature, pressure, and composition are all from Example 1.

State specification

Components	Methane	Hydrogen Sulfide
Mole fraction	0.5	0.5
Temperature	187.0 K	
Pressure	4.052 MPa	

```
▸ [0.5, 0.5]
```

```
begin
    model = EPPR78(["methane", "hydrogen sulfide"])
    T = 187.0
    p = 4.052e6

    z = [0.5, 0.5]
end
```

2. Generate initial guesses

$$\ln K_i = \ln \frac{P_{c,i}}{P_i} + 5.373(1+\omega_i)\left(1 - \frac{T_{c,i}}{T}\right)$$

The Wilson approximation is based on the ideal solution approximation, and is structured to match pure component vapour pressure at $T_r = 0.7$ and $T_r = 1.0$. It relies on the **critical temperature and pressure** as well as the **acentric factor**, all easily obtainable properties of the pure components. While it generally performs quite well, especially for mixtures relevant to the petrochemical industry, it has very poor predictions when used with hydrogen.

To calculate Wilson K-factors we use the function supplied by Clapeyron:

```
wilson_k_values(model, p, T)
```

3. Minimise our objective function

Next we minimise our objective function, making sure we tell our solver to use automatic differentiation for the derivatives. We specify Newton's method in the call to `solve(problem, method())`. Note that we find the minima by unconstrained minimisation of tm, but when investigating the actual stability we return the value of tpd.

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

```
function chemical_stability_analysis(model, p, T, z)
    Kʷ = wilson_k_values(model, p, T)
    z = z ./ sum(z)
    # Generate initial guesses
    w_vap = normalize(z ./ Kʷ, 1) # vapour-like
    w_liq = normalize(Kʷ .* z, 1) # liquid-like
    w0_vec = [w_liq, w_vap]

    # Objective function - Unconstrained formulation in mole numbers
    φ(x) = fugacity_coefficient(model, p, T, x)
    d(x) = log.(x) .+ log.(φ(x))
    d_z = d(z)

    tm(W) = 1.0 + sum(W .* (d(W) .- d_z .- 1.0))
    f(W, _) = tm(exp.(W))

    # Solve for our liquid and vapour guesses
    optf = OptimizationFunction(f, Optimization.AutoForwardDiff())
    prob(w0) = OptimizationProblem(optf, log.(w0))
    sol(w0) = solve(prob(w0), Newton())
    # Evaluate solutions
    sol_vec = sol.(w0_vec)

    # Extract minimum
    tm_min, idx = findmin([s.minimum for s in sol_vec])
    # W = exp.(sol_vec[idx].minimizer)
    tm_xmin = normalize(exp.(sol_vec[idx].u), 1)

    # Evaluate tpd
    tpd(x) = sum(x .* (log.(x) + log.(φ(x)) .- d_z))
    tpd_min = tpd(tm_xmin)

    return tm_xmin, tpd_min
end
```

```
([0.936465, 0.0635347], -0.120647)
```

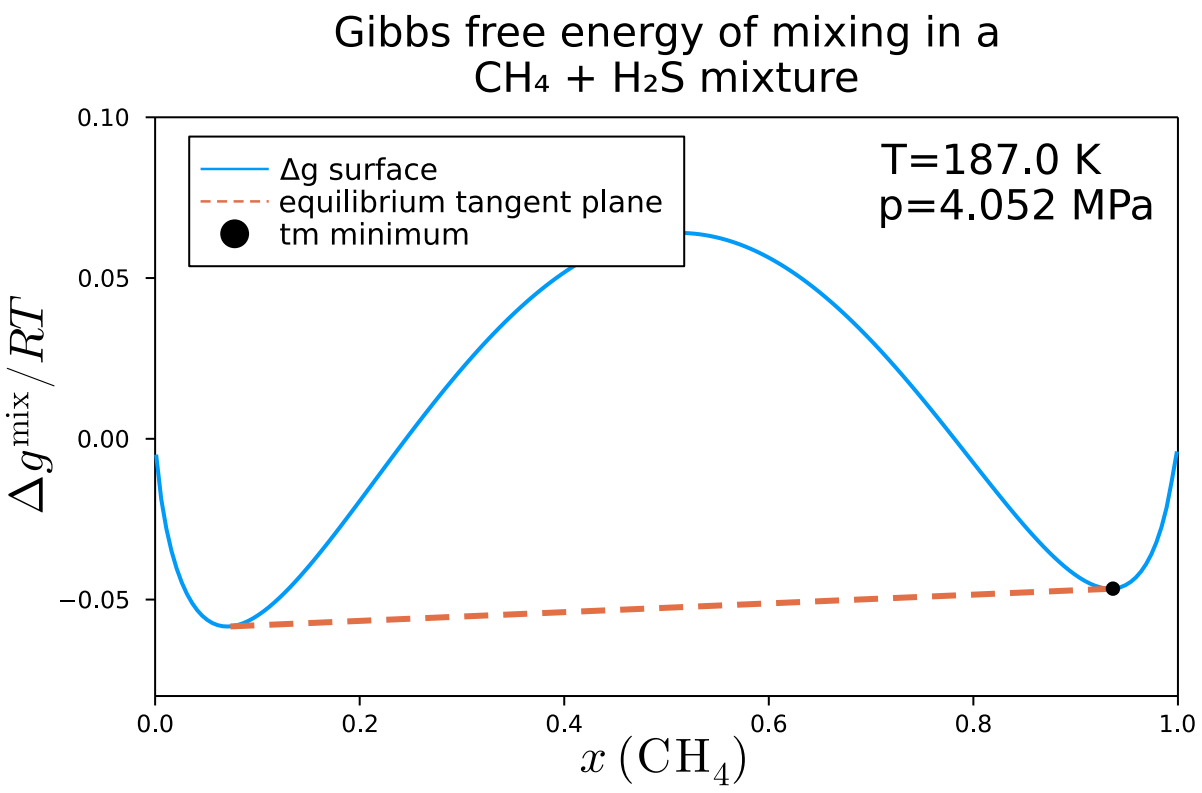
```
tm_xmin, tpd_min = chemical_stability_analysis(model, p, T, z)
```

```
(2×2 Matrix{Float64}: , 2×2 Matrix{Float64}: , -2.76719)
  0.0718775  0.928122    0.0362445  0.468009
  0.935469   0.0645309   0.463756   0.0319909
```

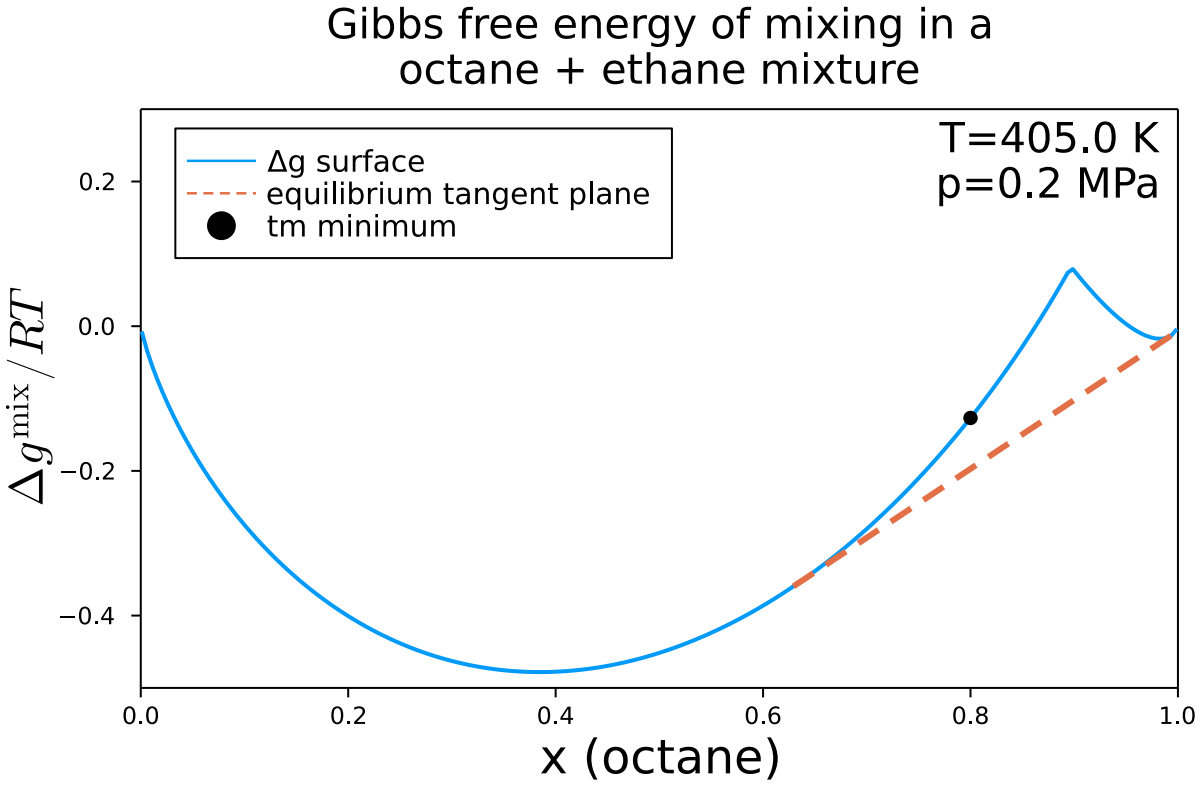
```
tp_flash(model, p, T, z)
```

We can see that our value of tpd_min, -0.1206, is less than 0. This correctly suggests that our mixture is unstable and a phase split will occur. On top of this, if we plot the minimum point we see it's incredibly close to the actual value of the equilibrium values - this secondary function of providing initial guesses to the flash solver is part of what makes stability analysis via the Gibbs tangent plane so powerful.

In certain cases, this can provide incredibly good initial guesses



Though this is, unfortunately, not the norm. Otherwise, we often obtain pretty good initial guesses, though not perfect. For example, with an octane + ethane mixture

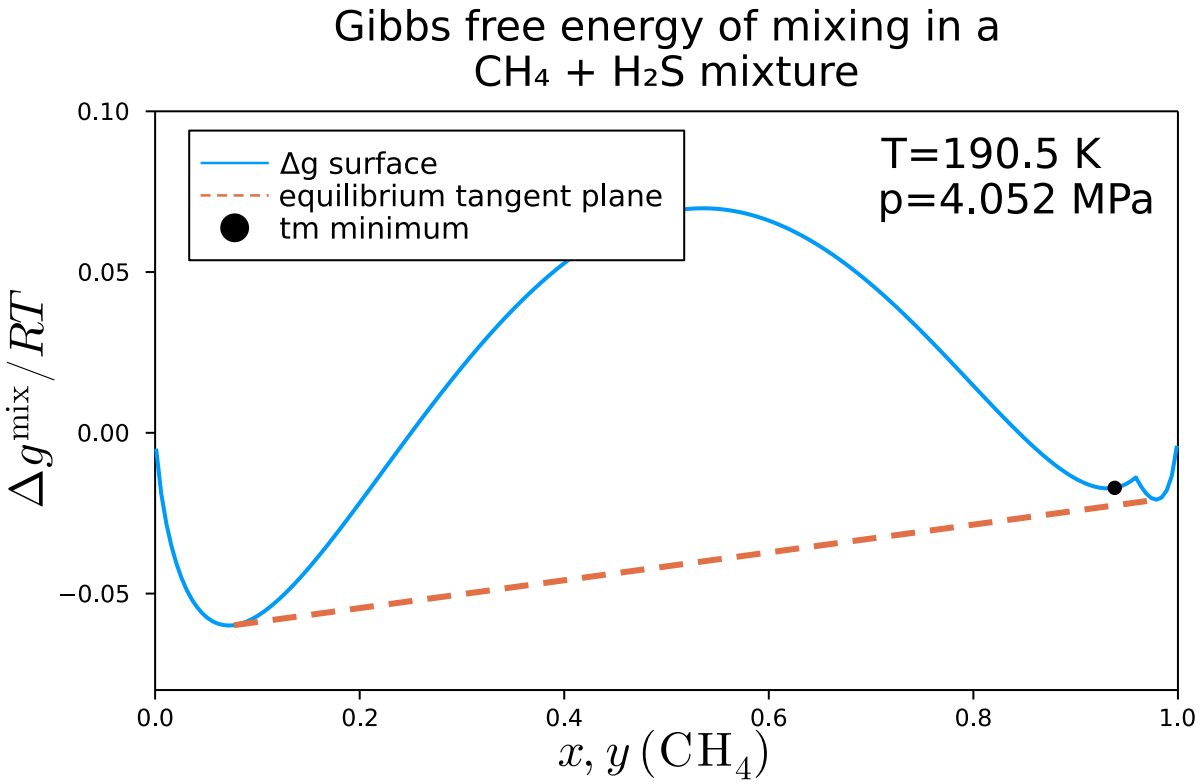


Problems and difficulties

However, our stability analysis algorithm is not foolproof. If we begin to increase the temperature from our state above, we approach the critical point of Methane and a third phase begins to appear. You can see this below using our Newton algorithm we converge to the incorrect minima.

We now have two minima very close to one another, and converging to the "correct" point can become very difficult without using more expensive optimisation techniques.

▸ ([0.938194, 0.061806], -0.116056)



Techniques for more robust stability analysis include:

- Multistart - here we start from a vapour-like and a liquid-like initial guess, but it would be possible to generate many more starting points through permutations to our initial guess or randomly generated points.
- Stochastic methods - evolutionary algorithms are typically quite good at finding global minima, but come with considerable expense.
- Tunnelling - Tunnelling algorithms have found considerable use in stability analysis, notably used in the HELD algorithm **[3]**

I

[1

M
M
L.
Th
Is
Fl
Pl
Pa
I.
St
Fl
Pl
Ec
(1
10
ci
by
84

[2

[3

P
Fr
E.
Ja
G
G
A
A
C
S.
Tl
H
A
Fc
M
M
Ec
Cc
W
G
Ec
O
St
C
&
C
Er
(2
10
ci
by
21