Pluto notebook for testing the `BandPyrometry.jl` module

Package `BandPyrometry.jl` contains methods to obtain the real surface's temperature from its thermal emission spectrum using multiwavelenght or band-pyrometry method.

Installation

To run this notebook, you need:

1. Install `julia` language itself from its official download page
2. Install Pluto notebook from `julia` REPL by entering the following commands line-by-line:

```
import Pkg
Pkg.add("Pluto")
using Pluto
Pluto.run()
```

The last line will launch the Pluto starting page in your default browser

3. Copy the entire GitHub repository to your local folder
4. Open this notebook file `BandPyrometry_test_git.jl` in `Pluto` by providing the full path to the *"Open a notebook"* text field on `Pluto`'s starting page. As far as `Pluto` has its own package manager, it will automatically install all necessary dependancies, which are marked in `using` cell of this file .

Table of Contents

Part I. Introduction

I.I. The `blackbody` vs `real surface` thermal emission

All heated bodies emit thermal radiation. According to Planck's law, the spectrum of ideal emitter (called the *blackbody*) is governed solaly by its temperature. The blackbody spectral intensity can be calculated as follows

$$I_{blackbody}(\lambda,T)=\frac{C_1}{\lambda^5}\cdot\frac{1}{e^{\frac{C_2}{\lambda T}}-1},$$

where C_1 = 1.191043e8, $W\cdot\mu m/m^2\cdot sr$ and C_2 = 14387.752, $\mu m\cdot K$, λ - wavelength in μm, T - temperature in Kelvins

A real surface thermal emission intensity is lower than the one of the blackbody. The fraction of blackbody thermal radiation intensity emitted by a real surface is characterized by directional spectral emissivity $\epsilon(\lambda,T,\vec{\Omega})$:

$$I_{real\ \ surface}(\lambda,T,\vec{\Omega})=\epsilon(\lambda,T,\vec{\Omega})\cdot I_{blackbody}(\lambda,T),$$

here $\vec{\Omega}$ stays for direction.

It is interesting that, unlike the blackbody, the real surface thermal emission (in general) depends on the direction of radiation. Therefore, the most general characteristic for thermal radiation of a real surface is the `directional spectral emissivity`.

In Planck.jl module there are several functions to calculate the blackbody thermal emission spectra (and various derivatives, integrals etc.). The following figure show the impact of spectral emissivity on the real surface thermal emission intensity.

Blackbody spectral range, μm :

λ_{left} = 0.1 _ 25.0 λ_{right}

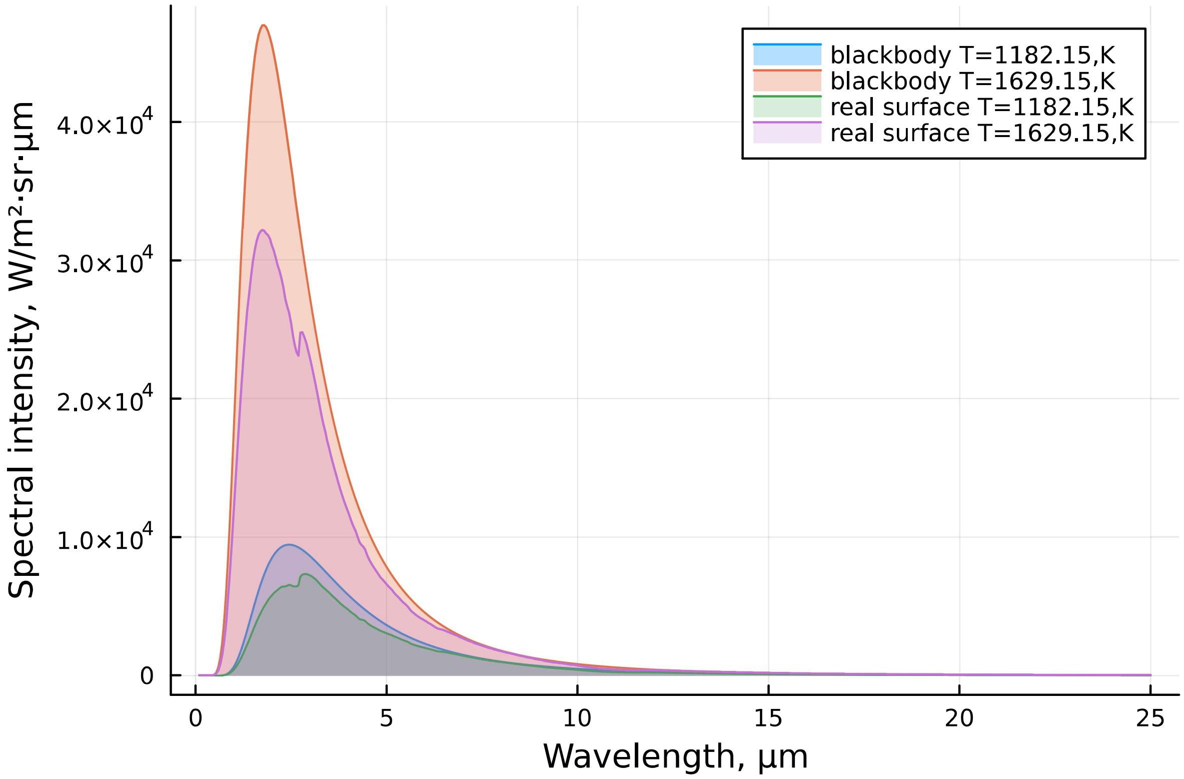
Blackbody temperatures:

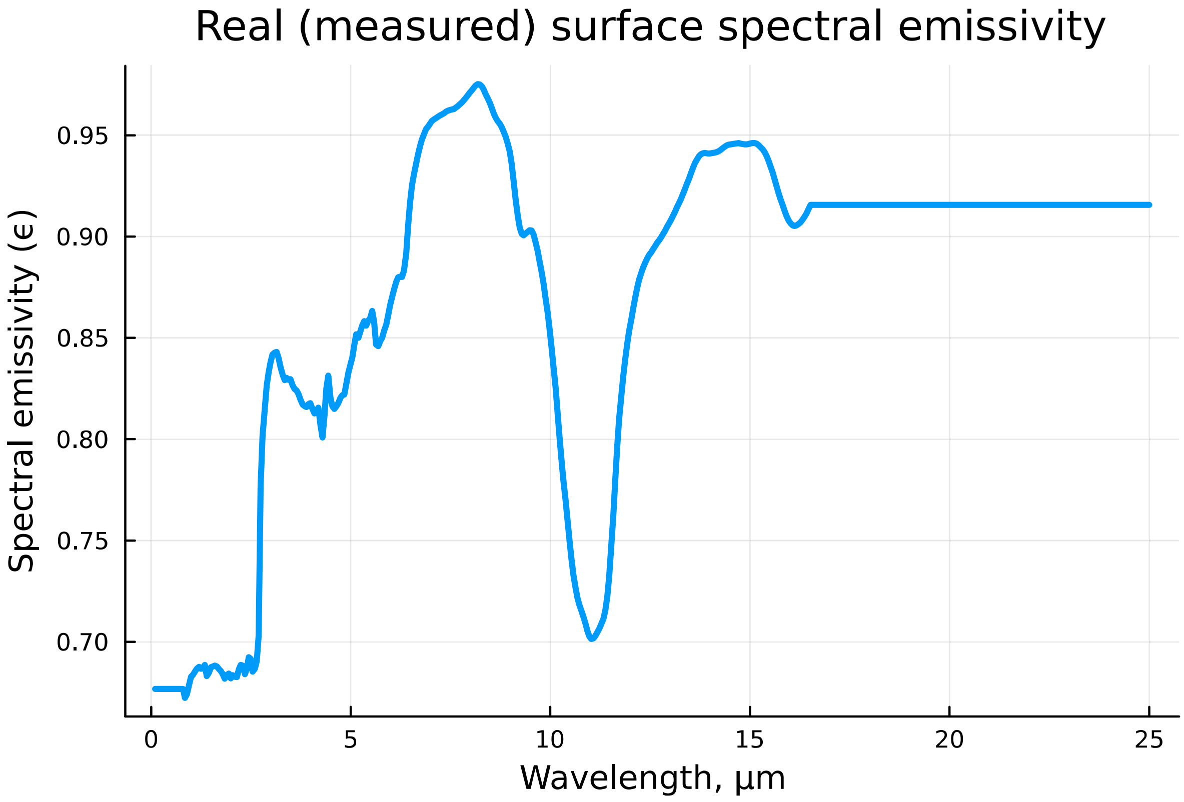
T_1 = 909.0 ^oC

T_2 = 1356.0 ^oC

Scales :

xscale = identity ⌄ yscale identity ⌄





I.II. Partial radiation pyrometry

As far as the `blackbody` thermal radiation energy strongly depends on temperature, this quantity can be used to measure the temperature of a real surface. This is the general idea of partial radiation pyrometry: **measure intensity to get the temperature**. As far as the intensity is a directional quantity, a pyrometer needs collimating optics (a telescope).The real surfaces emissivity often varies sufficiently with the wavelength, at the same time, partial radiation pyrometers assume constant emissivity (so-called `grey`-band approximation). Thus, for industrial purposes, it is useful to have several pyrometers, each working within a relatively narrow spectral band. In the spectral range of a partial radiation pyrometer, emissivity should not vary significantly to make the assumption of constant emissivity relevant.

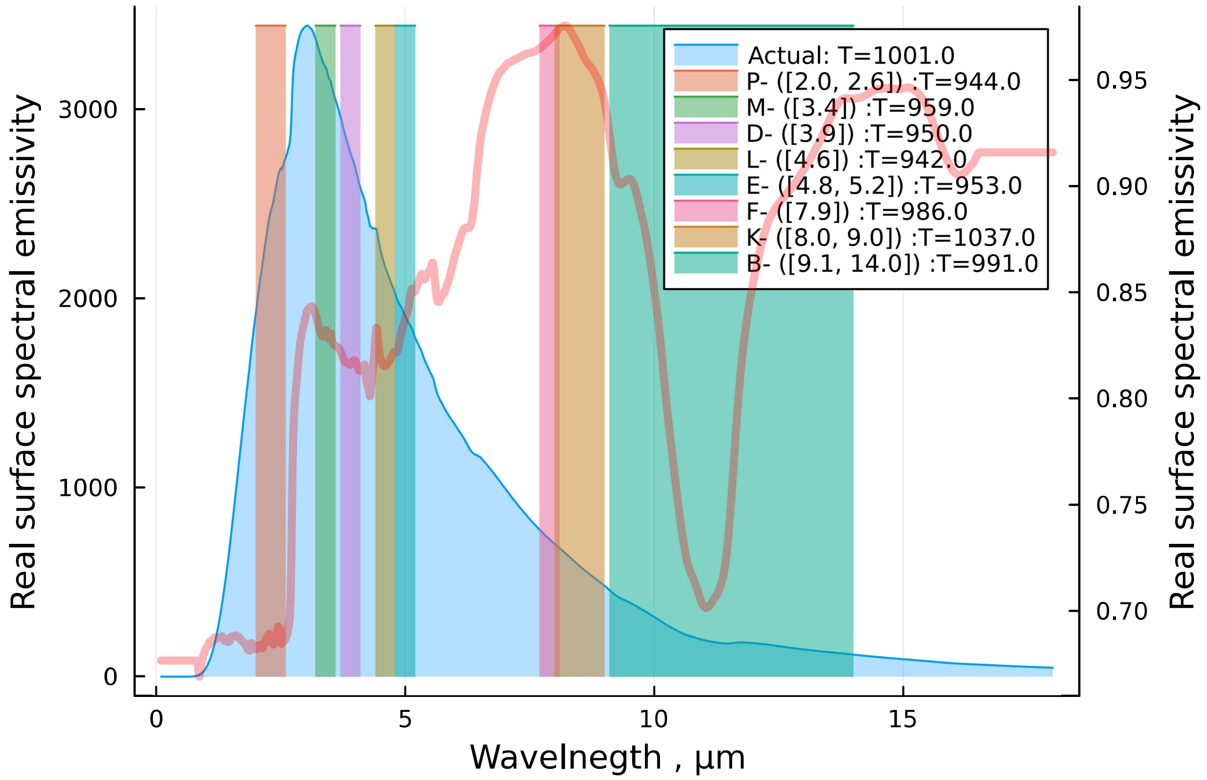
One of BandPyrometry.jl modules Pyrometers.jl provides several function to work with `virtual` partial radiation pyrometers.

Table of different \`virtual pyrometer\` types provied by \`Pyrometers.jl\` and corresponding wavelength regions

	B	M	P	D	L	E	K	F
	Float64	Float64	Float64	Float64	Float64	Float64	Float64	Float64
λ_l	9.1	3.4	2.0	3.9	4.6	4.8	8.0	7.9
λ_r	14.0	#undef	2.6	#undef	#undef	5.2	9.0	#undef

It is interesting to look how various pyrometers (with their emissivity set to one) `measure` the temperature of a real surface. The following figure shows the real surface thermal emission intensity and several common pyrometers types working regions. In the legend their `measured` temperature is shown. The `mesured` temperature for each pyrometer type is obtained by fitting the blackbody power to the real surface power both integrated over pyrometer's working spectral range.

Real surface temperature 1000.6806806806807



For this particular material the type `-F` pyrometer readings are closer to the real temperature, because of the real surface emissivity being closer to unity for this pyrometer's spectral region. All results are summarized in the following table.

Table of temperatures `measured` by different pyrometers with the spectral emissivity settled to unity (T_0) and to the calculated grey-ϵ (T_1) the real temperature is T_r=1000.6806806806807

type	λ region,μm	T_0 (ϵ=1),K	grey-ϵ	T_1 (grey-ϵ),K
P	[2.0, 2.6]	944.0	0.687101	1001.0
M	[3.4]	959.0	0.830863	1001.0
D	[3.9]	950.0	0.816117	1001.0
L	[4.6]	942.0	0.81506	1001.0
E	[4.8, 5.2]	953.0	0.861575	1001.0
F	[7.9]	986.0	0.968791	1001.0
K	[8.0, 9.0]	1037.0	1.06864	1001.0
B	[9.1, 14.0]	991.0	0.984471	1001.0

I.III. Multiwavelength pyrometry

Unlike classical partial radiation pyrometry, which requires setting a constant emissivity in some relatively narrow wavelength region, the **multiwavelength pyrometry** allows one to obtain the temperature of a surface without knowing the emissivity. More about multiwavelength pyrometry can be found e.g. in Multi-spectral pyrometry—a review

In order to achieve this goal, the **multiwavelength pyrometry** assumes that the dependence of emissivity on wavelength in some spectral range can be described by some relatively small number (N) of parameters. Hence, if you have measured thermal radiation intensity at relatively large number (M) of wavelengths, and if $M > N+1$, you can formulate the optimization problem in space of $N+1$ optimization variables viz $\vec{x} = [a_0, \ldots, a_{N-1}, T]$, here $[a_0, \ldots, a_{N-1}]$ are N emissivity approximation coefficients, and the $(N+1)$'th optimization variable T is the temperature. The **multiwavelength pyrometry** optimization problem has several features that can be utilized in order to obtain a computationally-effective algorithm:

- First, the emissivity approximation is a linear problem, this means that emissivity approximation coefficients can be taken independent of both the optimization variables and the independent variables (wavelength)
- Second, a highly non-linear term (viz Planck function) depends on only one of the optimizaiton variables
- And third, the target function is the product of linear and non-linear optimization problems

Mathematical consequencies of these features are described in this repository supplementary materials download pdf in more details.

In `BandPyrometry.jl` , the real surface thermal emission is approximated as a product of Planck function (ideal surface thermal emission) and linear (with respect to the optimization variables e.g. polynomial coefficients) approximation of spectral emissivity.

In `BandPyrometry.jl` the spectral emissivity is approximated as a linear combination of basis functions:

$$\epsilon(\lambda) = \sum_{n=0}^{N-1} a_n \cdot \phi_n(\lambda)$$

where ϕ_n is the basis function column vector, e.g. for standard basis it is:

$$\phi_{\mathbf{n}}(\lambda) = \begin{bmatrix} \lambda_1{}^{\mathrm{n}} \\ \vdots \\ \lambda_M{}^{\mathrm{n}} \end{bmatrix}$$

Full emission spectrum of a real surface is calculated as:

$$\mathbf{I_{real\ surface}(\lambda, T) = I_{blackbody}(\lambda, T) \cdot \epsilon(\lambda) = I_{blackbody}(\lambda, T) \cdot \sum_{n=0}^{N-1} a_n \cdot \phi_n(\lambda)}$$

Now, the optimization problem can be formulated:

$$\vec{x}^* = argmin\{F(\vec{x})\}$$

$$F(\vec{x}) = \sum_{i=1}^{M}[y_i - I_{blackbody}(\lambda_i, T) \cdot (\sum_{n=0}^{N-1} a_n \cdot \phi_n(\lambda_i))]^2$$

$$\vec{x} = [\vec{a}, T]^t$$

where $\vec{a}$ is the column vector of emissivity approximation ($[]^t$ stays for transposition), $\vec{x}^*$ is the local minimum

To solve this optimization problem `BandPyrometry.jl` package provides functions to evaluate the discrepancy function F, ∇F and $\nabla^2 F$ which are needed to solve the optimization problem using zero, first or second order optimization algorithms. It also has special type to work with the emissivity linear approximation.

I.IV. Emissivity approximation functions

To approximate the emissivity `BandPyrometry.jl` uses several polynomial bases:

- Standard basis (from `Polynomials.jl` package)
- Chebyshev basis (from `Polynomials.jl` package)
- Legendre polynomials (from `LegendrePolynoials.jl` package)
- Trigonometric basis: $\phi_n(\lambda) = sin(\pi \cdot n\lambda)$ for odd n, and $\phi_n(\lambda) = cos(\pi \cdot n\lambda)$ for even n

All basis vectors are stored in a structure called `VanderMatrix`, this type:

1. Stores basis vectors for selected polynomial type and degreee - columns of matrix V: $V = \left[\vec{\phi}_1, \ldots, \vec{\phi}_n\right]$
2. The resulting emissivity can be calculated as a product of `VanderMatrix` and the vector of emissivity approximation polynomial coefficients vector: $\vec{\epsilon} = V \cdot \vec{a}$

Independent variable vector λ is stored in normalized (in order to fit withing the range of -1...1) form, `VanderMatrix` also stores all data needed to return the original λ vector. Matrix V can be used to fit (in a least-square sense) by solving the overdetermined system of equations: $\vec{a} = V^{\dagger} \cdot \vec{\epsilon}$, where []^{\dagger} means pseudo-inverse. The package implements this in `polyfit` function (show docs) ☐.

Emissivity fitting spectral range, μm :
λ_{left} = 4.0 _ 8.0 λ_{right}
Submit

Select the polynomial type = leg

Set polynomial degree : 3 ⌄ (the polynomial degree = numer of basis functions - 1, thus, zero order polynomial is an all-units vector)

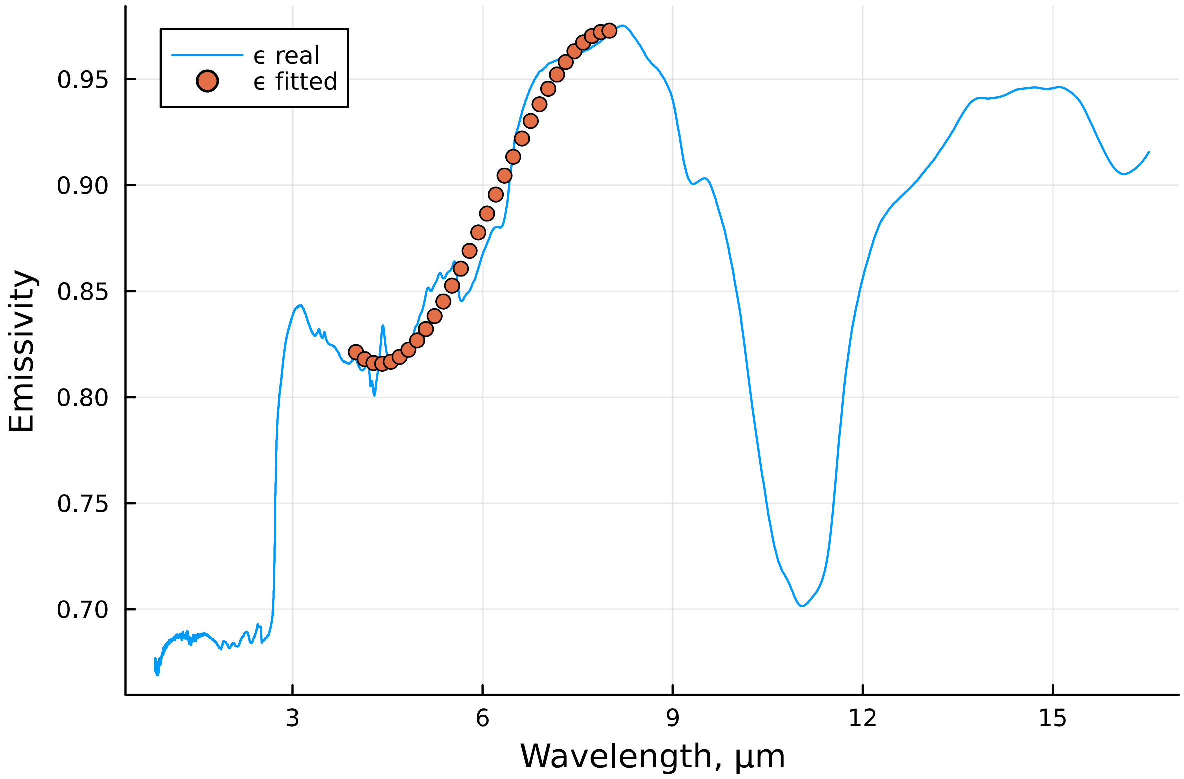


Table of the coefficients of emissivity linear approximation in the band from 4.0 to 8.0 µm using leg bases type, the goodness of fit = 0.06201053374738862

	Polynomial coefficients
a0	0.887127
a1	0.0970663
a2	0.00995443
a3	-0.0212695

Part II. `BandPyrometry.jl` testing

In this notebook, the "measured" thermal emission spectrum is calculated as the same model as the `BandPyrometryPoint.jl` internal representation, further this spectrum is fitted using optimization tools provided by **`Optimization.jl`** package, some random noise can be added `(optionally)`.

II.I. "Measured" emissivity generation

Measured spectrum fitting region:

λ_{min} ,µm = 8.01

λ_{max} ,µm = 13.0

Submit

Select the polynomial type = leg Submit

Set the "measured" spectrum emissivity approximation coefficients:

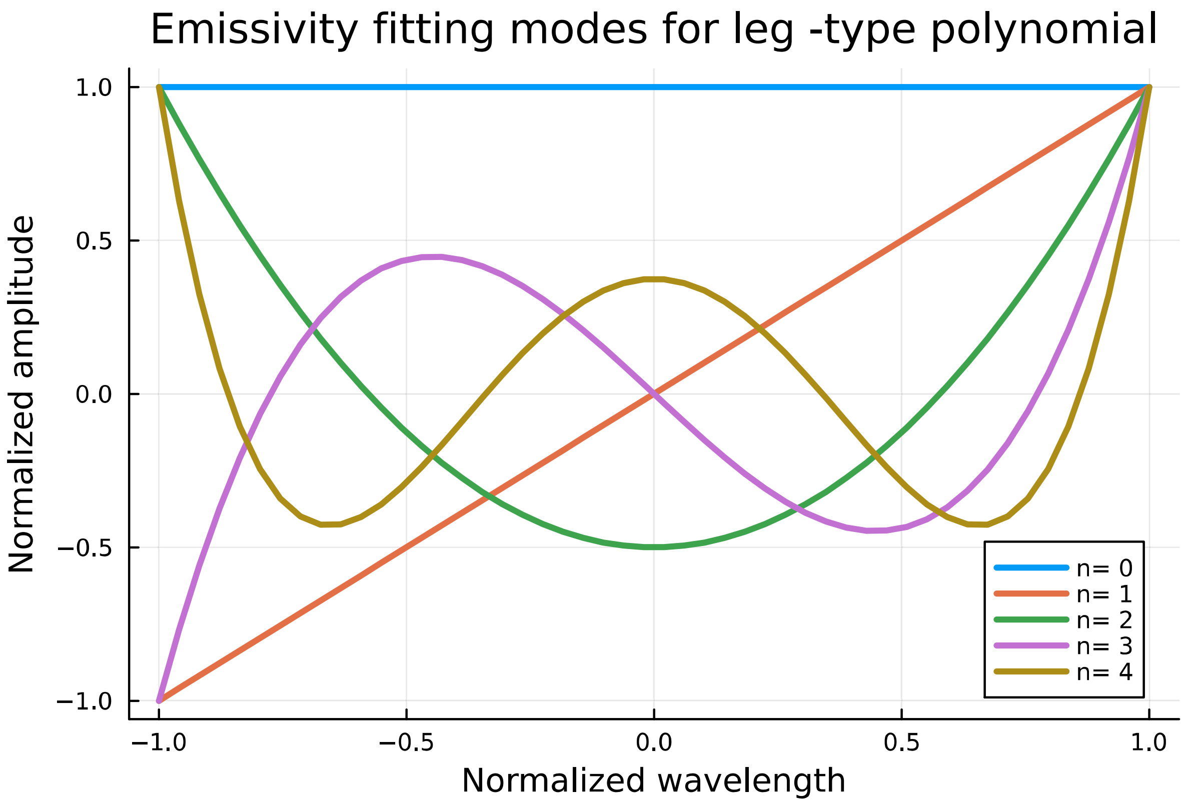
a0 = 0.3
a1 = 0.1
a2 = 0.2
a3 = 0.2
a4 = 0.2
a5 = 0.2
Submit

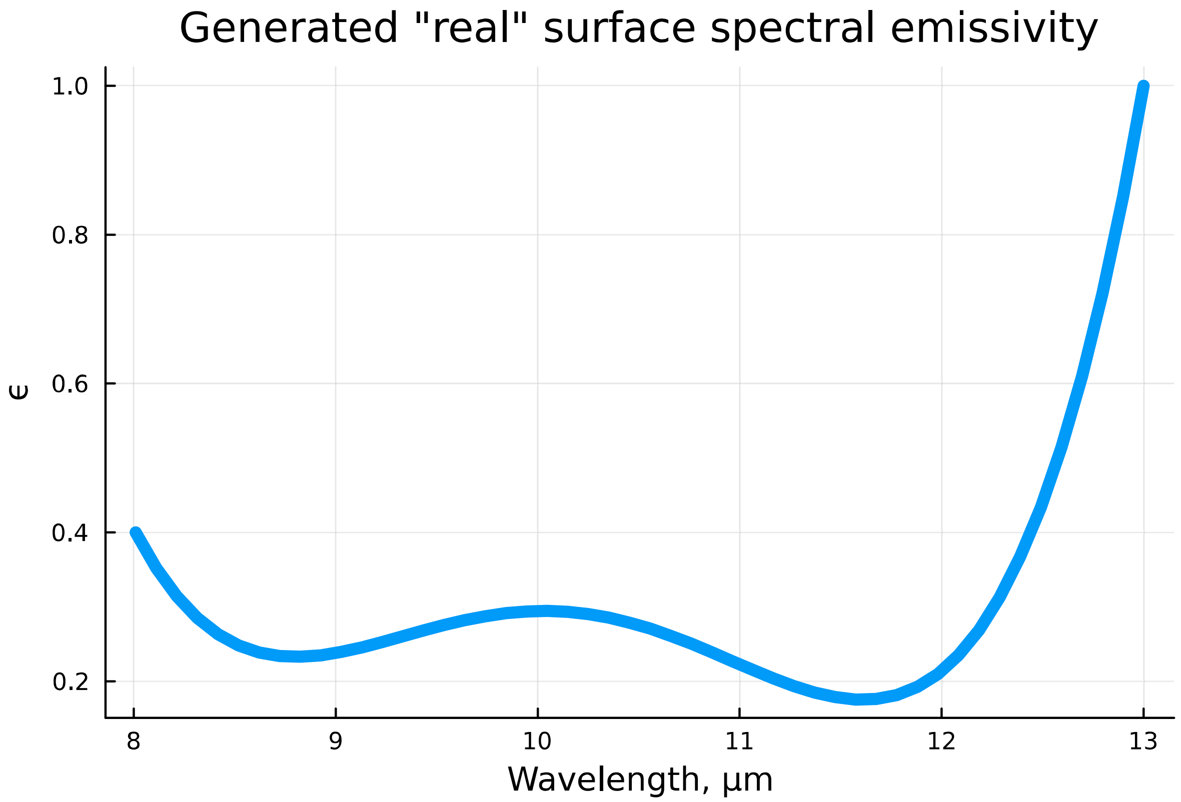
Set polynomial degree : 4 Submit (the polynomial degree= numer of basis functions-1, thus zero order polynomial is constant)

The following two figures show:

1)basis vectors for selected polynomial (columns of `VanderMatrix.v`): V

2)the resulting emissivity, calculated as a product of `VanderMatrix` and the vector of emissivity approximation polynomial: $\vec{\epsilon} = \left[\vec{\phi}_1, \dots, \vec{\phi}_n\right] \cdot \vec{a} = V \cdot \vec{a}$





II.II. Generating the "measured" spectral intensity

"Measured" spectrum temperature, K = 1000.5275275275276 Submit

Real values of the optimization variables for ”measured” thermal emission spectrum:

$$[0.3, 0.1, 0.2, 0.2, 0.2, 1000.5275275275276]$$

add noise to the "measured" spectrum = 0.0 Submit

II.III. Solving the optimization problem

Choose the optimization method NelderMead Submit

unconstraint Submit

Selected optimization method is NelderMead the actual optimizer is Optim.NelderMead

```
(T = 1000.53, a = [0.299997, 0.0999995, 0.199998, 0.199999, 0.199998], ϵ = StaticArraysCor
```

```
11.349917 seconds (2.74 M allocations: 159.038 MiB, 1.87% gc time, 99.31%
compilation time)
```

Goodness of fit:

$$\left[\begin{array}{lll} T_{fitted} = 1000.5320962721028 & T_{real} = 1000.5275275275276 & \Delta T/T = 4.566335 \\ |\vec{a}_{fitted}| = 0.46903782502472674 & |\vec{a}_{real}| = 0.469041575982343 & |\vec{a}_{fitted} - \vec{a}_{real}| = 3.768 \end{array}\right.$$

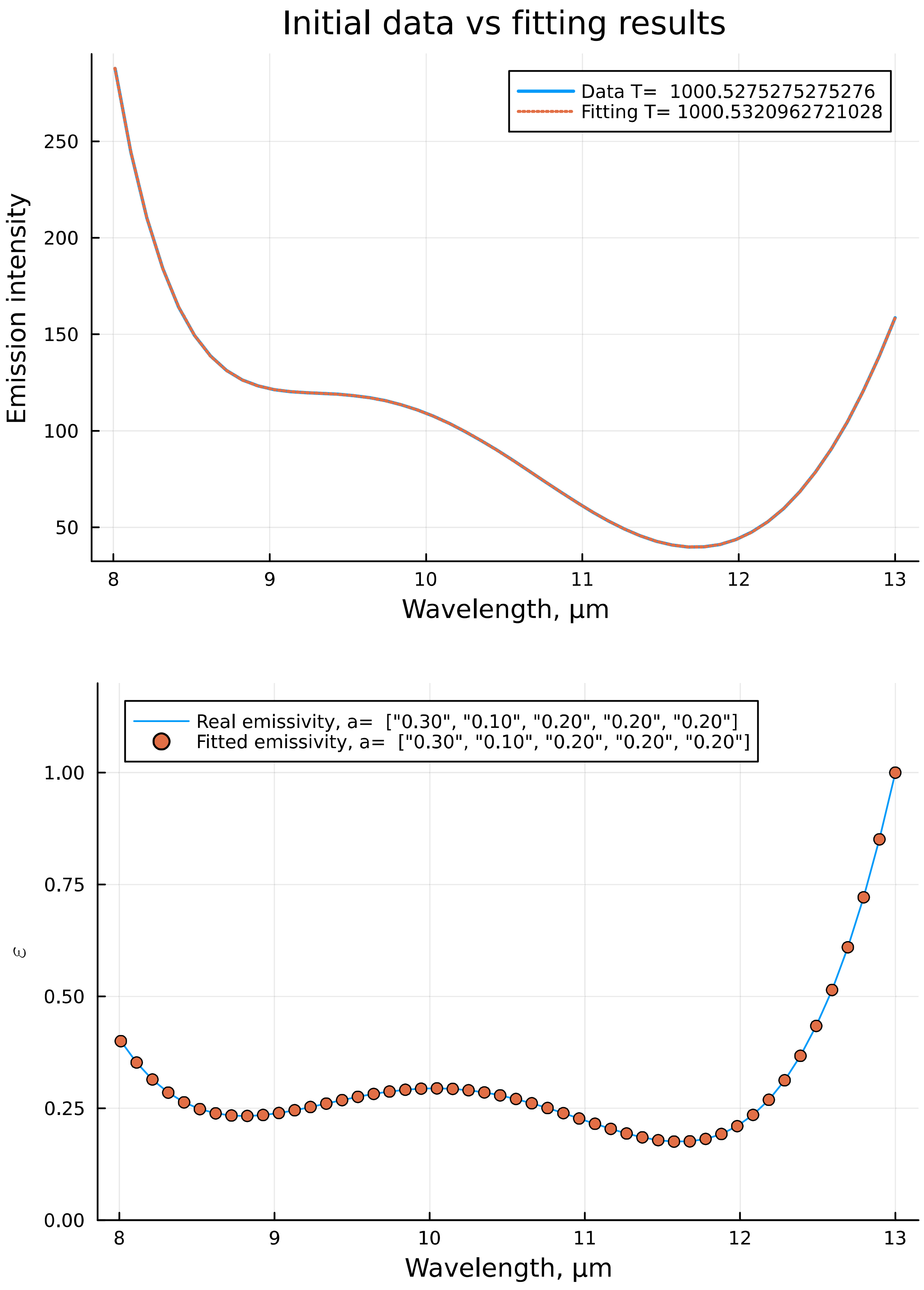
```
 * Status: success

 * Candidate solution
    Final objective value:     6.577496e-09

 * Found with
    Algorithm:     Nelder-Mead

 * Convergence measures
    √(Σ(yᵢ-ȳ)²)/n ≤ 1.0e-08

 * Work counters
    Seconds run:   0  (vs limit Inf)
    Iterations:    828
    f(x) calls:    1318
```



Final discrepancy value: 6.577496415006803e-9