```julia
using Plots,StaticArrays,Polynomials,Interpolations, RecipesBase, LegendrePolynomials, PlutoUI, LinearAlgebra, PrettyTables, Optimization,OptimizationOptimJL
```

```julia
include(raw"PolynomialWrappers.jl")
```

Table of Contents

```julia
PlutoUI.TableOfContents(indent=true, depth=4, aside=true)
```

Bernstein polynomial for constraint optimization

Introduction

A common problem in constrained nonlinear optimization is converting nonlinear constraints applied to some function of the optimization variables into box constraints on the optimization variables themselves. Simply, converting nonlinear constraints to box constraints is necessary.

For example, in multiwavelength pyrometry, the measured signal can be approximated as a product of two functions: one (spectral emissivity) is linear with respect to the optimization variables, while the other (`blackbody` thermal emission spectrum) is nonlinear.

$\vec{b}^* = argmin\{F(\vec{b})\}$ - optimization problem

$F(\vec{b}) = \sum_{i=1}^{M}[y_i - I_{bb}(\lambda_i, T) \cdot (\sum_{k=0}^{n} a_k \cdot \phi_k(\lambda_i))]^2$ - discrepancy function

$\vec{b} = [\vec{a}, T]^t$ - optimization variables vector, T is the temperature, $\vec{a}$ - coefficients of emissivity approximation, $I_{bb}(\lambda_i, T)$ - blackbody thermal emission spectrum, y_i - measured intensity.

$\epsilon(\lambda) = \sum_{n=0}^{N-1} a_n \cdot \phi_n(\lambda)$ - spectral emissivity is a linear combination of some basis functions $\phi_n(\lambda)$

There is a physical constraint on the emissivity, which follows from the fact that real object cannot emit more radiation than the blackbody:

$\epsilon(\lambda) \in (0...1)$ for the whole spectral range.

Sometimes, the region of emissivity variation can be narrowed; for example, it may be known that, for a particular material, the emissivity lies within the range of $[\epsilon_a \dots \epsilon_b]$. The question is how to convert the emissivity variation region to the emissivity approximation coefficients $\vec{a}$ variation.

System of inequalities of contraints on emissivity:

$\vec{\epsilon_a} \leq \vec{\epsilon} \leq \vec{\epsilon_b}$ (vectors in independent variables space R^M)

Should be somehow converted to inequality constraints on emissivity approximation variables:

$\vec{a_a} \leq \vec{a} \leq \vec{a_b}$ (vector in optimization variables space R^N !)

This problem can be solved using the Bernstein polynomial basis.

Matrix form of polynomial approximation

Each polynomial basis has a set of basis functions (monomials):

$[\phi_0(x), \dots, \phi_n(x)]$

E.g. for standard polynomial basis: $\phi_k = x^k$

Columns of Vandermonde matrix (V) are the polynomial basis functions evaluated at coordinates x, thus number of columns in V is equal to the `degree of polynomial + 1`:

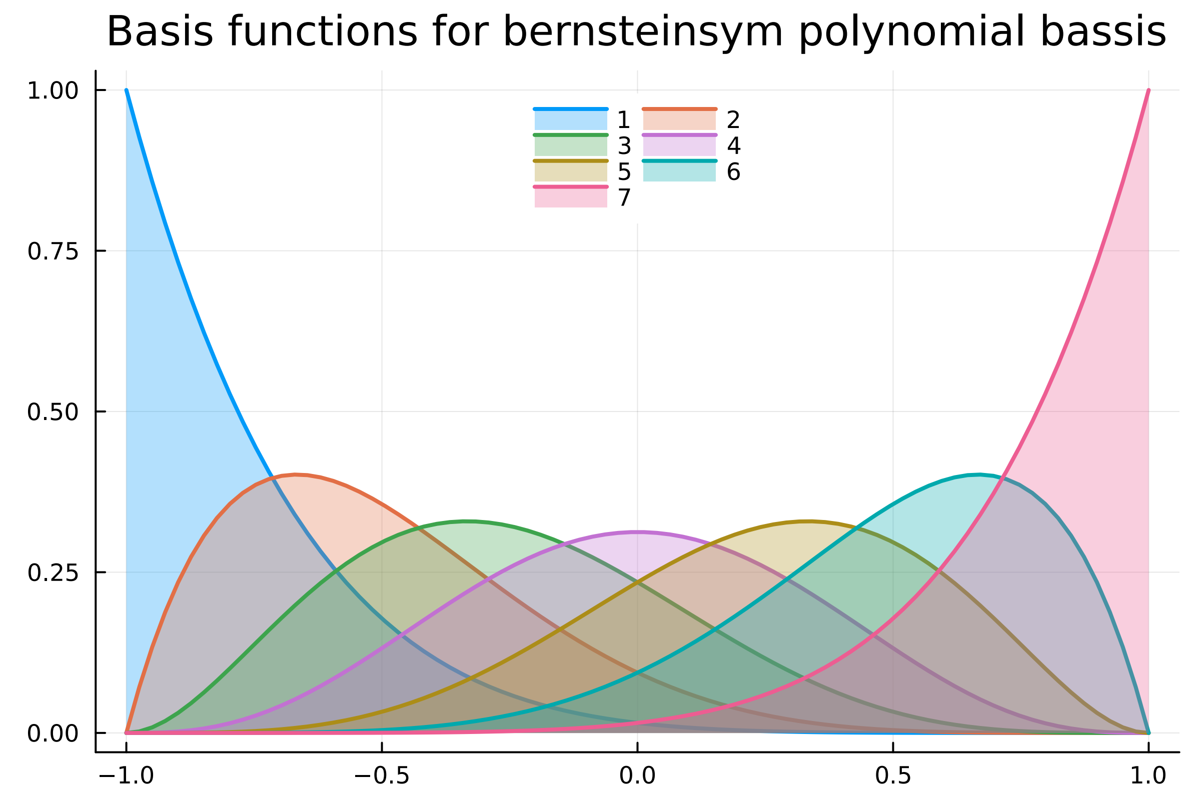
$V = \left[\vec{\phi}_0, \dots, \vec{\phi}_n\right]$ - Vandermonde matrix

Select polynomial basis type for the initial function bernsteinsym ⌄

Initial function polynomial degree 6 ⌄

Show infilled ? ☑

The following figure shows basis functions (Vandermonde matrix columns) for polynomial of type **bernsteinsym**



Bernstein polynomial basis

Bernstein polynomial basis of degree n has the following set of $n+1$ basis functions (monomials):

$\beta_k^n(x) = \binom{n}{k}x^k(1-x)^{n-k}$ - Bernstein basis function for $x \in [0..\,.1]$, $k \in [0...n]$.

$\binom{n}{k} = \frac{n!}{(n-k)!k!}$

For general case, when $x \in [a, b]$

$\beta_k^n(x) = \binom{n}{k}(\frac{x-a}{b-a})^k(\frac{b-x}{b-a})^{n-k}$ - Bernstein basis function for $x \in [a...b]$, $k \in [0...n]$

In the following, all formulas will be provided for $x \in [0..\,.1]$ basis, because of simplicity, but for calculations, symmetric Bernstein (**bernsteinsym** in selection dropdown) basis ($x \in [-1..\,.1]$) was used because of the ransge $[-1..\,.1]$ is more natural for other polynomial bases, like Legendre and Chebyshev polynomials

Main features of B-polynomial basis:

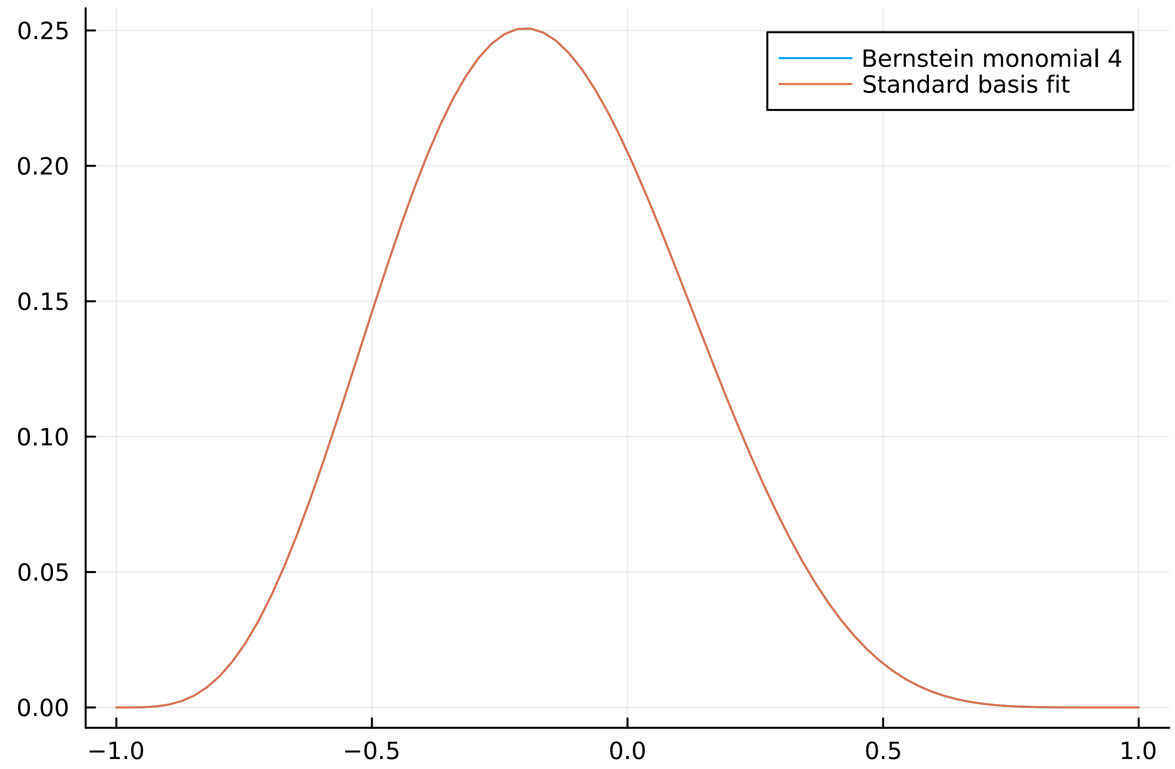
1. Each B-monomial $\beta_k^n(x)$ is a polynomial of degree n.

Roughly speaking, all B-monomial have the same `units`

Standard basis polynomial fitting

a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	
0.205078	-0.410156	-0.615234	1.64063	0.410156	-2.46094	0.410156	1.64063	-0.615234	-0.

The following table and figure show 4 ⌄ 'th B-monomial for B-basis of degree 10 fitting using standard basis polynomial



2. Each B-monomial has a single maximum with the value and location governed by n and k.

B-basis function $\beta_k^n(x)$ maximum value is $max(\beta_k^n(x)) = k^k \cdot n^n \cdot (n-k)^{n-k} \cdot \binom{n}{k}$. It is located at coordinate: $argmax(\beta_k^n(x)) = \frac{k}{n}$

3. All B-monomials are positive for all x

4. The summation over all B-monomials within any B-basis set gives one for any coordinate x:

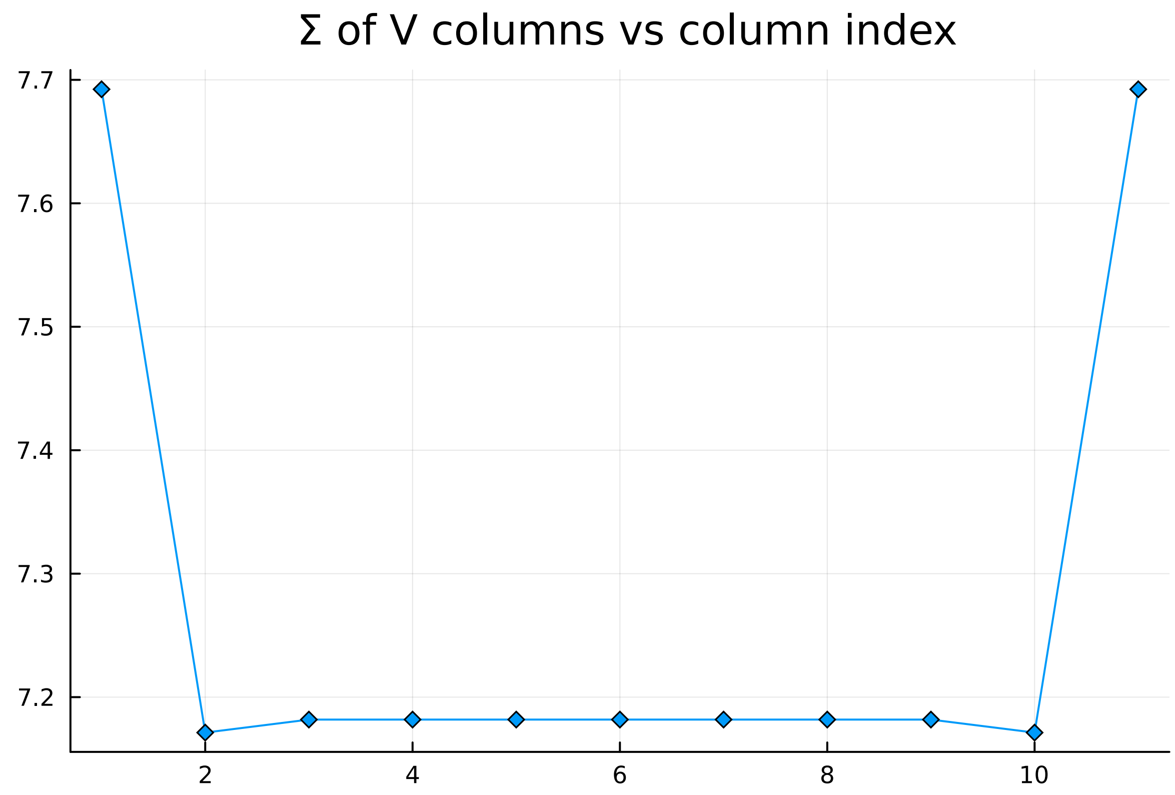
$\Sigma_{k=0}^{n} \beta_k^n(x) = 1$.

The third property means that each row of Vandermonde matrix sums to one: $\Sigma_i V[i,:] = 1$, hence

$\Sigma_{i,j} V[i,j] = M$, M is the number of rows (size of x vector)

$\Sigma_{i,j} V[i,j] =$ 79.99999999999989

Is the summation of B Vandermonde matrix over all values for x with 80 points



Now, consider the expantion of function $f(x)$ in Bernstein polynomial basis of degree n:

$f(x) = \Sigma_{k=0}^{n} a_k \beta_k^n(x)$

$\vec{f} = V\vec{a}$ - here $\vec{f}$ is the vector of function values evaluated at $\vec{x}$ coordinates. (the least-square solution of this systemof equations is $\vec{a} = V^\dagger \vec{f}$, where $V^\dagger$ is pseudo-inverse)

Bernstein polynomial approximation coefficients:

Bernstein polynomial fitting

Coordinate	Bernstein coefficient	Function value	delta/F, %
-1.0	0.299	0.299	2.78484e-13
-0.8	0.179	0.216752	17.417
-0.6	0.159	0.185925	14.4816
-0.4	0.172333	0.180528	4.53939
-0.2	0.188048	0.185006	-1.64419
0.0	0.196619	0.191184	-2.84297
0.2	0.199	0.195719	-1.6765
0.4	0.199	0.19805	-0.47981
0.6	0.199	0.198852	-0.0744148
0.8	0.199	0.198995	-0.00269605
1.0	0.199	0.199	1.39475e-14

Coefficients of f(x) function, $f(x) = \Sigma_{k=0}^{n} a_k \phi_k^n$, where n=6 and ϕ_k^n are the basis functions for bernsteinsym basis

a_0 = 0.299

a_1 = 0.099

a_2 = 0.199

a_3 = 0.199

a_4 = 0.199

a_5 = 0.199

a_6 = 0.199

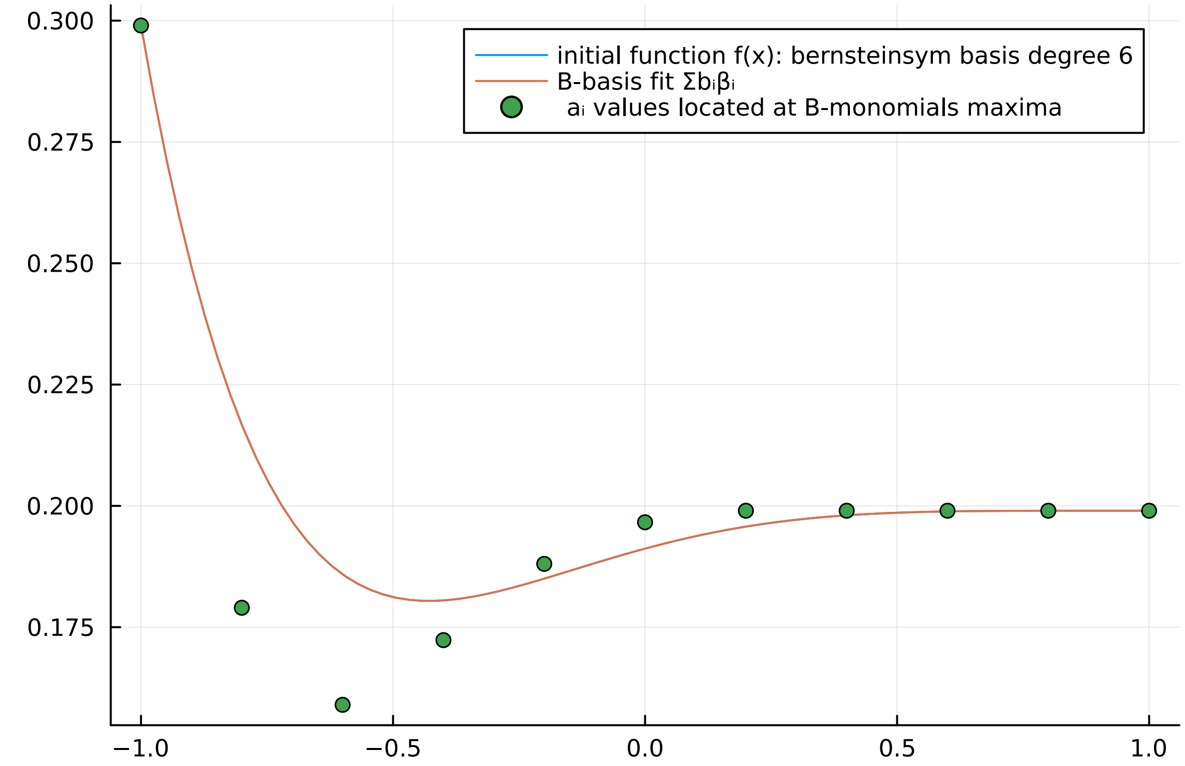
a_7 = 0.199

a_8 = 0.199

a_9 = 0.199

a_{10} = 0.199

Show B-basis fitting ☑



Fix axes limits ☐

y_{left} = -0.01 y_{right} = 1.0

Add noise 0.0

Figure shows the following plots:

1. Initial function $f(x)$ (which is summation of monomials of basis bernsteinsym of degree 6)
2. The result of $f(x)$ fitting using B-polynomial basis: $\Sigma_{k=0}^{n} a_k \beta_k^n$
3. Scattered points are the values of a_k vs the locations of corresponding monomials β_k^n

Bernstein fitting basis degree 10

It is clearly visible that the values of B-polynomials coefficients tend to the values of function itself!

The oposite is also true, if the initial function is a linear cobination of B-monomials, than the value of function in the vicinity of the corresponding monomial maxima location can be adjusted by adjusting the value of the coefficient (to check - set the polynomial bassis type of the initial function to **bernsteinsym** and use sliders)

Bernstein polynomials for constraints mapping

Now, we are going to the main topic - converting a nonlinear constraints to the box-constraints.

The problem is the following:

Fit the polynomial function $f(\vec{a},x)$ to the measured data $y(x)$ with the following constraints: $f(\vec{a},x) \leq q(x)$ and $f(\vec{a},x) \geq g(x)$, where $\vec{a}$ is the the optimization variables vector, x is the independent variable, $g(x)$ and $q(x)$ are known nonlinear functions.

The loss function will be a least-square discrepancy:

$\Phi(\vec{a}) = \Sigma_i(y(x_i) - f(\vec{a},x_i))^2$ - loss function

In mathematical form this gives:

$\vec{a^*} = argmin(\Phi(\vec{a}))$

Where the feasible region of $\vec{a}$ is a subject to the following constraints:

$f(\vec{a},x) \leq q(x)$

$f(\vec{a},x) \geq g(x)$

We will be looking for the function $f(\vec{a},x)$ as an expantion in Bernstein basis:

$f(\vec{a},x) = \Sigma_j a_j \beta_j^n$ or in matrix form: $\vec{f} = V_\beta \cdot \vec{a}$, here V_β is Bernstein basis Vendermonde matrix.

We also expand both $q(x)$ and $g(x)$ in Bernstein of the same degree:

$\vec{g} = V_\beta \cdot \vec{a}_{lower}$

$\vec{q} = V_\beta \cdot \vec{a}_{upper}$

Both $\vec{a}_{lower}$ and $\vec{a}_{upper}$ are known vectors (as far as $g(x)$ and $q(x)$ are known functions)

Original problem can now be reformulated in Bernstein basis:

$\vec{a^*} = argmin(\Phi(\vec{a}))$

Bernstein polynomial optimization with constraints

Coefficients of f(x) function

a_0 = 0.297

a_1 = 0.099

a_2 = 0.201

a_3 = 0.201

Add noise 1.0

Coefficients of f(x) function lower boundary g(x), $g(x) \leq f(x)$

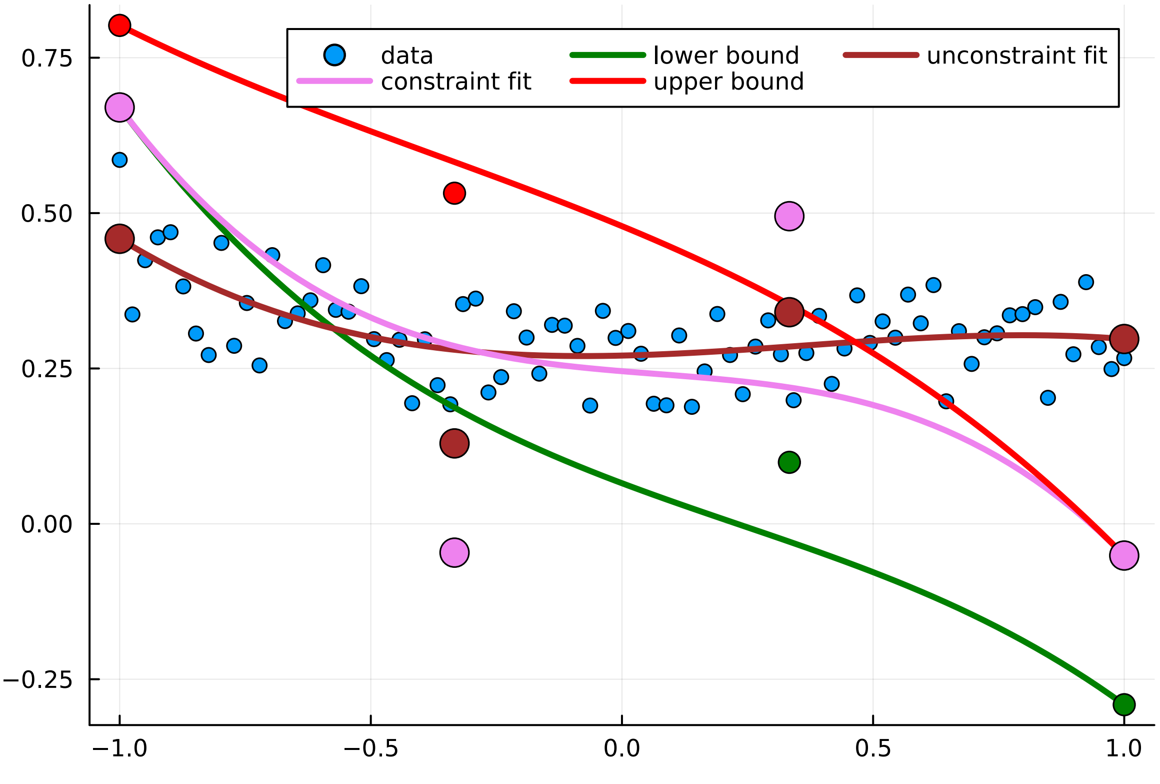
a_0 = 0.67

a_1 = -0.051

a_2 = 0.099

a_3 = -0.291

The following figure shows scattered initial data, and four curves together with four sets of markers plot for B-polynomials coefficients. Each color corresponds to the same curve.



Coefficients of f(x) function upper boundary q(x), $f(x) \leq q(x)$

a_0 = 0.802

a_1 = 0.532

a_2 = 0.495

a_3 = -0.051

Constraint optimization

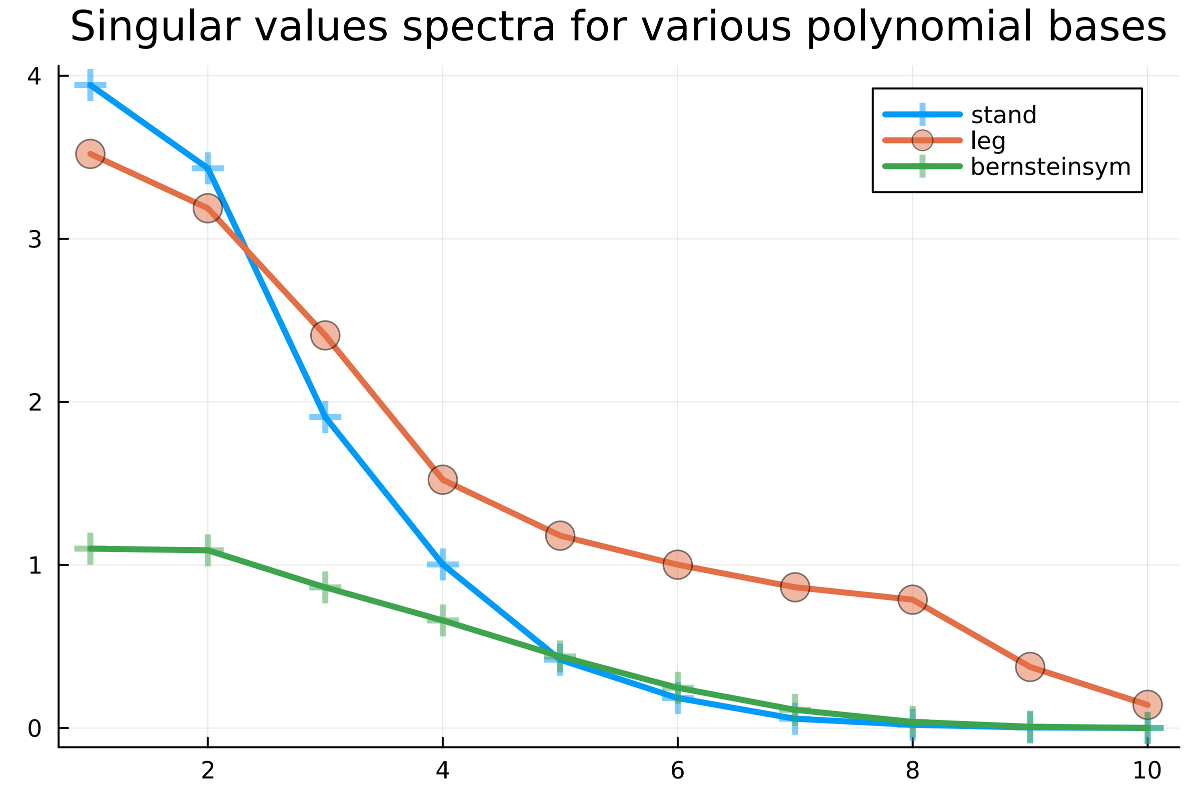
Data coeffs	Lower bound	Unconstraint fit	Constraint fit	Upper bound
0.297	0.67	0.458653	0.67	0.802
0.099	-0.051	0.129432	-0.0463159	0.532
0.201	0.099	0.34052	0.495	0.495
0.201	-0.291	0.297408	-0.051	-0.051

Select optimizer Particle swarm ⌄

```
SciMLBase.OptimizationSolution{Float64, 1, Vector{Float64}, OptimizationBase.OptimizationCache{
```

```
(0.67, -0.051, 0.099, -0.291)
```

```
fit_res =
  (StaticArraysCore.SVector{7, Float64}: [0.191188, 0.028125, -0.0234375, -0.03125, 0.0703125,
```


Singular values spectra for various polynomial bases
stand
leg
bernsteinsym
0
1
2
3
4
2
4
6
8
10

a = **10**