Taylor Series

By: Nova White

Goals

- This Julia open package focuses on Taylor polynomial expansions in one or more independent variables.
- It is a general-purpose software package that includes topics such as: Julia, automatic-differentiation, polynomials, and taylor-expansions.

$$f_k(\mathbf{x}_0) = \sum_{m_1+\cdots+m_d=k} f_{m_1,\ldots,m_d} \, (x_1 - x_{0_1})^{m_1} \cdots (x_d - x_{0_d})^{m_d} = \sum_{|\mathbf{m}|=k} f_{\mathbf{m}} \, (\mathbf{x} - \mathbf{x}_0)^{\mathbf{m}}.$$

References

- The software's copyright is held by Luis Benet and David P. Sanders
- The package is licensed under MIT "Expat"
  - It is free of charge for anyone obtaining the copy of this software
- Anyone can contribute by reporting an issue, fixing an existing issues, adding new functionality to improve speed, or updating documentation.
  - 21 contributors and counting
  - Communication is done through github issues/pull requests
- This software is beneficial for those studying higher level physics and mathematics.

Taylor Series: One Independent Variable

- Taylor expansions in one variable are represented by the Taylor1 type, which consists of a vector of coefficients and the maximum order considered for the expansion.
    - Taylor1([1, 2, 3],4) # Polynomial of order 4 with coefficients 1, 2, 3
- If the length of the vector does not correspond with the order, order is used, which effectively truncates polynomial to degree order.

```
[1]: using Pkg; Pkg.add("TaylorSeries")
```

```
[14]: using TaylorSeries

      #Example 1: Taylor series in one variable

      t = Taylor1(Float64, 5)

      exp(t)
```

```
[14]:  1.0 + 1.0 t + 0.5 t² + 0.16666666666666666 t³ + 0.041666666666666664 t⁴ + 0.008333333333333333 t⁵ + 𝒪(t⁶)
```

```
[15]: using TaylorSeries

      #Example 2: Taylor series in one variable

      t = Taylor1(Float64, 5)

      log(1 + t)
```

```
[15]:  1.0 t - 0.5 t² + 0.3333333333333333 t³ - 0.25 t⁴ + 0.2 t⁵ + 𝒪(t⁶)
```

Taylor Series: Multivariate

- A polynomial in N>1 variables can be represented in (at least) two ways: As a vector whose coefficients are homogeneous polynomials of fixed degree, or as a vector whose coefficients are polynomials in $N-1$ variables.
  - In the package they used the first implementation.
- The implementation in the open source imposes the user to specify the (maximum) order considered and the number of independent variables at the beginning, which can be conveniently done using set_variables.
  - x, y = set_variables("x y")

[17]:
```
#Example 3: Multivariate Taylor series

x, y = set_variables("x y", order=2);

exp(x + y)
```

[17]: $1.0 + 1.0\ x + 1.0\ y + 0.5\ x^2 + 1.0\ x\ y + 0.5\ y^2 + \mathcal{O}(\|x\|^3)$

Taylor Series: Differential and Integral Calculus

- We can also define taylor expansions with N+1 variables where one of the variables is somewhat special
    - This kind of expansions are of interest when studying the dependence of parameters, for example when considering the dependence of the solution of a differential equation on the initial conditions, around a given solution

[17]:
```
#Example 3: Multivariate Taylor series

x, y = set_variables("x y", order=2);

exp(x + y)
```

[17]: $1.0 + 1.0\ x + 1.0\ y + 0.5\ x^2 + 1.0\ x\ y + 0.5\ y^2 + \mathcal{O}(\|x\|^3)$

[22]:
```
#Example 4: Differential and integral calculus on Taylor series:
x, y = set_variables("x y", order=4);

p = x^3 + 2x^2 * y - 7x + 2
```

[22]: $2.0 - 7.0\ x + 1.0\ x^3 + 2.0\ x^2\ y + \mathcal{O}(\|x\|^5)$

[23]:
```
∇(p)
```

[23]: 2-element Vector{TaylorN{Float64}}:

$- 7.0 + 3.0\ x^2 + 4.0\ x\ y + \mathcal{O}(\|x\|^5)$

$2.0\ x^2 + \mathcal{O}(\|x\|^5)$

[24]:
```
integrate(p, 1)
```

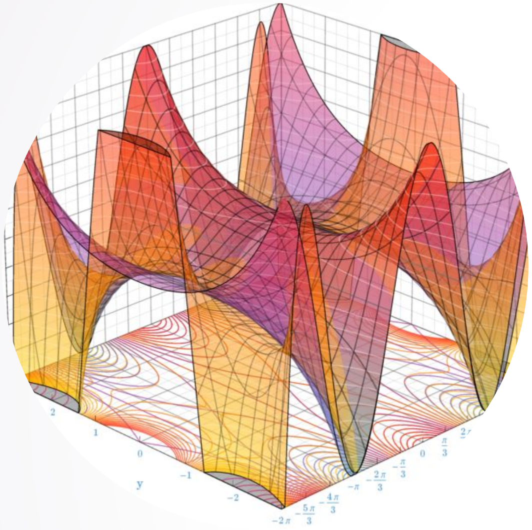
[24]: $2.0\ x - 3.5\ x^2 + 0.25\ x^4 + 0.6666666666666666\ x^3\ y + \mathcal{O}(\|x\|^5)$

[25]:
```
integrate(p, 2)
```

[25]: $2.0\ y - 7.0\ x\ y + 1.0\ x^3\ y + 1.0\ x^2\ y^2 + \mathcal{O}(\|x\|^5)$

Question:

"It says the software started off under python, but it is currently in Julia. Why Julia? What makes it a better source when calculating taylor series?"



Experiment

One experiment we can carry out with this software is observing how the Julia functions behave when handling negative values