Yale NUS College

Developing Mathematical Software for Efficient Representation of Functions and Numerical Integration: julia and ApproxFun

Dion Ho
05 September 2018

The Schrodinger Equation

6 The time-dependent linear Schrodinger equation

The time-dependent linear Schrodinger equation is a complex partial differential equation used to model quantum systems with wavefunctions which change with time.

The problem is given as follows:

Partial Differential Equation, PDE: $[\partial_t + i(-i\partial_x)^2]q(x,t) = q_t - iq_{xx} = 0$

Initial Condition, IC: $q(x,0) = q_0(x)$

Boundary Condition, BC: $q_x(0,t) + \beta q(0,t) = h(t)$.

Solving this equation requires computing the **wavefunction:** $\Psi(x,t)$, though we denoted it $q(x,t)$ instead.

Formula for the wavefunction

The wavefunction

$$2\pi q(x,t) = \int_{-\infty}^{\infty} e^{i\lambda x - i\lambda^2 t}\hat{q}_0(\lambda)\,\mathrm{d}\lambda$$
$$+ \int_{\delta D^+} \frac{2i\lambda}{\beta + i\lambda}\left(e^{i\lambda x - i\lambda^2 t}\right)\int_0^{\tau} e^{i\lambda^2 s} h(s)\,\mathrm{d}s\,\mathrm{d}\lambda$$
$$+ \int_{\delta D^+} \frac{\beta - i\lambda}{\beta + i\lambda}\left(e^{i\lambda x - i\lambda^2 t}\right)\hat{q}_0(-\lambda)\,\mathrm{d}\lambda$$
$$- \int_{\delta D^+} \frac{\beta - i\lambda}{\beta + i\lambda}\left(e^{i\lambda x - i\lambda^2 t}\right)\left(e^{i\lambda^2 \tau}\right)\hat{q}(-\lambda;\tau)\,\mathrm{d}\lambda$$

Direct (Symbolic) Integration

- $\int x \, \mathrm{d}x = \frac{x^2}{2} + C$
- $\int \sin(x) \, \mathrm{d}x = -\cos(x) + C$
- $\int xe^x \, \mathrm{d}x = xe^x + \int e^x \, \mathrm{d}x = xe^x + e^x + C$
- $\int \frac{1}{x(x-1)} \, \mathrm{d}x = \int \left(\frac{1}{x-1} - \frac{1}{x} \right) \mathrm{d}x = \log|x-1| - \log|x| + C$

Direct Integration vs Numerical Integration

- Direct integration produces **EXACT VALUES.**
  - $\int x \, dx = \frac{x^2}{2} + C$
  - $\int \sin(x) \, dx = -\cos(x) + C$
  - $\int xe^x \, dx = xe^x + \int e^x \, dx = xe^x + e^x + C$
  - $\int \frac{1}{x(x-1)} \, dx = \int \left(\frac{1}{x-1} - \frac{1}{x} \right) dx = \log|x-1| - \log|x| + C$

- Numerical integration produces **APPROXIMATIONS.**

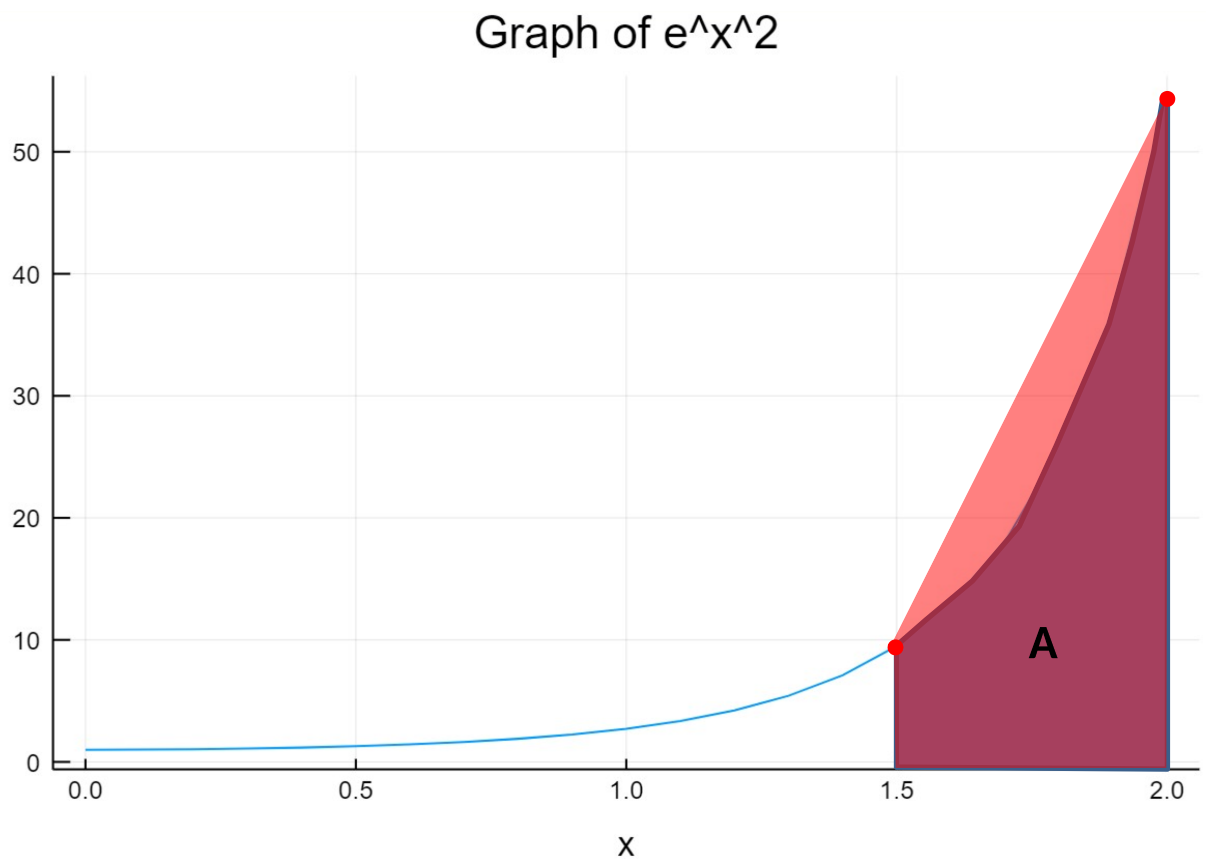
The Problem with Direct Integration

Direct integration is not feasible, or even possible, for every function.

- $\int \sqrt{\tan(x)}\,\mathrm{d}x$

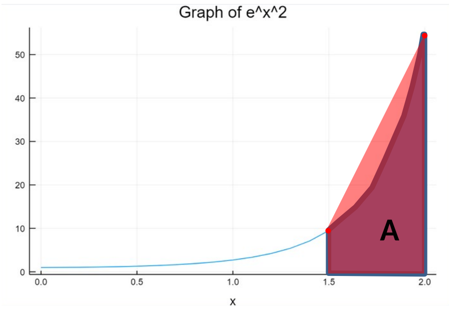
- $\int e^{x^2}\,\mathrm{d}x$

Integration as Area Under the Curve



$$\int_{1.5}^{2} e^{x^2} \, \mathrm{d}x$$

Newton-Cotes Rules



- **Trapezoidal rule:**

$$\int_b^a f(x)\,\mathrm{d}x \approx \frac{a-b}{2}(f_a + f_b)$$

- Simpson's rule:

$$\int_b^a f(x)\,\mathrm{d}x \approx \frac{a-b}{3}(f_a + 4f_c + f_b)$$

- Simpson's $\frac{3}{8}$ rule:

$$\int_b^a f(x)\,\mathrm{d}x \approx \frac{3(a-b)}{8}(f_a + 3f_c + 3f_d + f_b)$$

- Boole's rule:

$$\int_b^a f(x)\,\mathrm{d}x \approx \frac{2(a-b)}{45}(7f_a + 32f_c + 12f_d + 32f_e + 7f_b)$$

f_a denotes $f(a)$.

Polynomial Approximation

$$\int_b^a f(x)\,\mathrm{d}x \approx$$

Taylor Polynomial Approximation

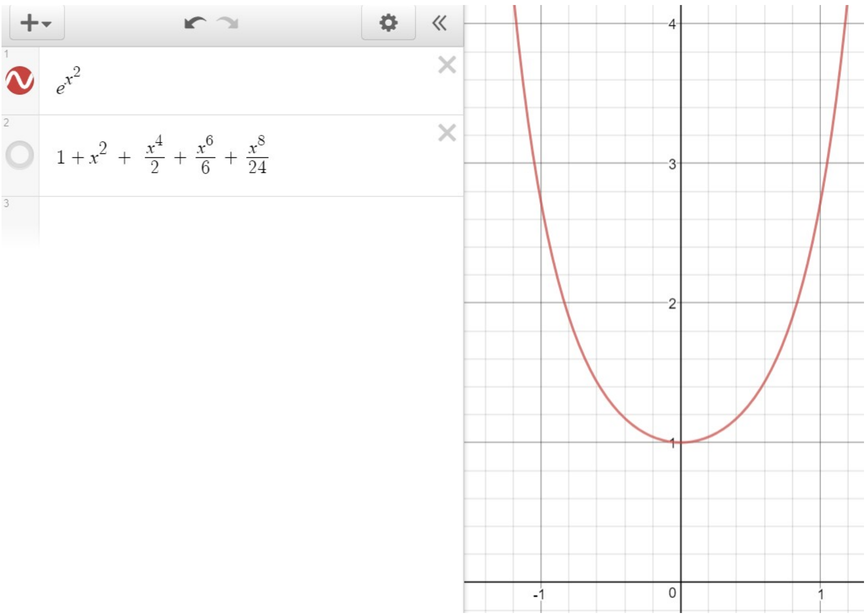
Taylor series expansion of $f(x)$ at number a:

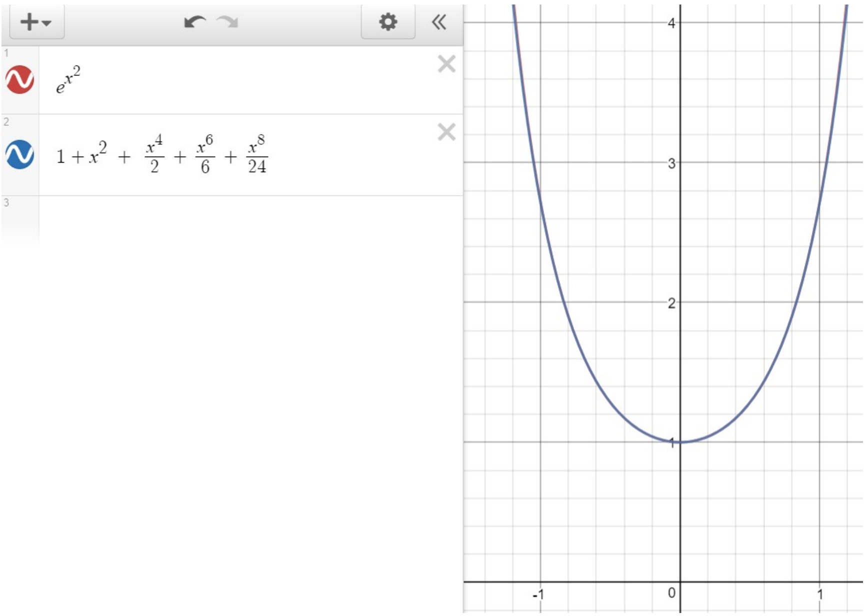
$$f(x) = f(a) + \frac{f'(a)}{1!}(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \dots$$

Truncation

For example:

Taylor Polynomial Approximation (example)





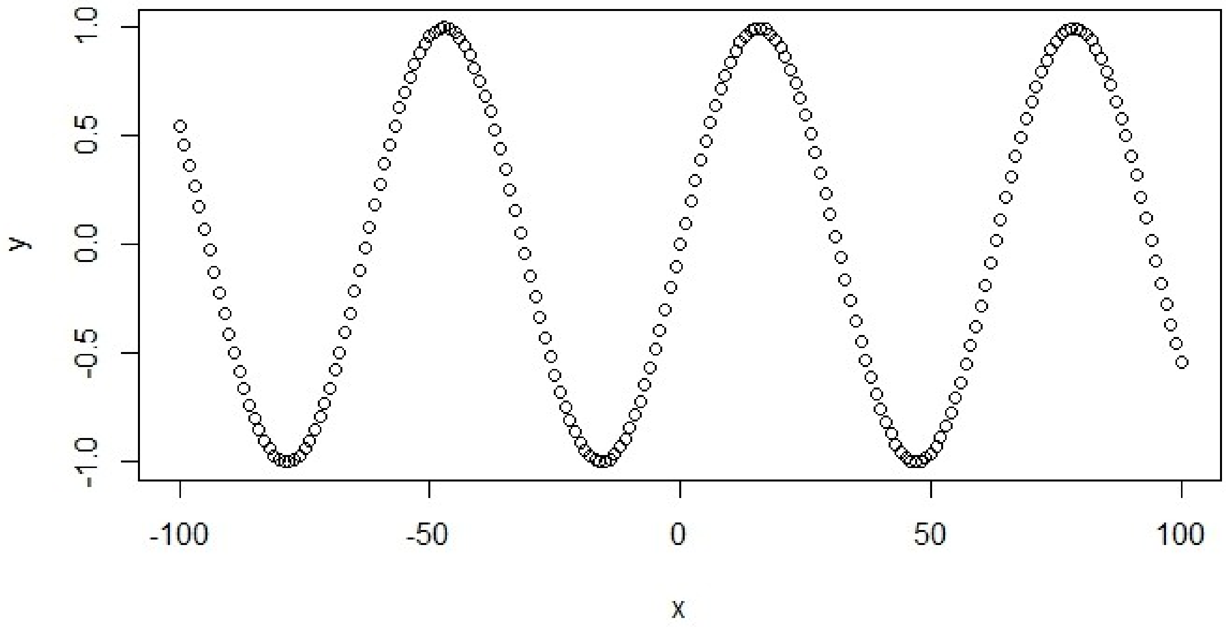
$$e^{x^2} \approx 1 + x^2 + \frac{x^4}{2} + \frac{x^6}{6} + \frac{x^8}{24}$$

Taylor Polynomial Approximation (example)

$$e^{x^2} \approx 1 + x^2 + \frac{x^4}{2} + \frac{x^6}{6} + \frac{x^8}{24}$$

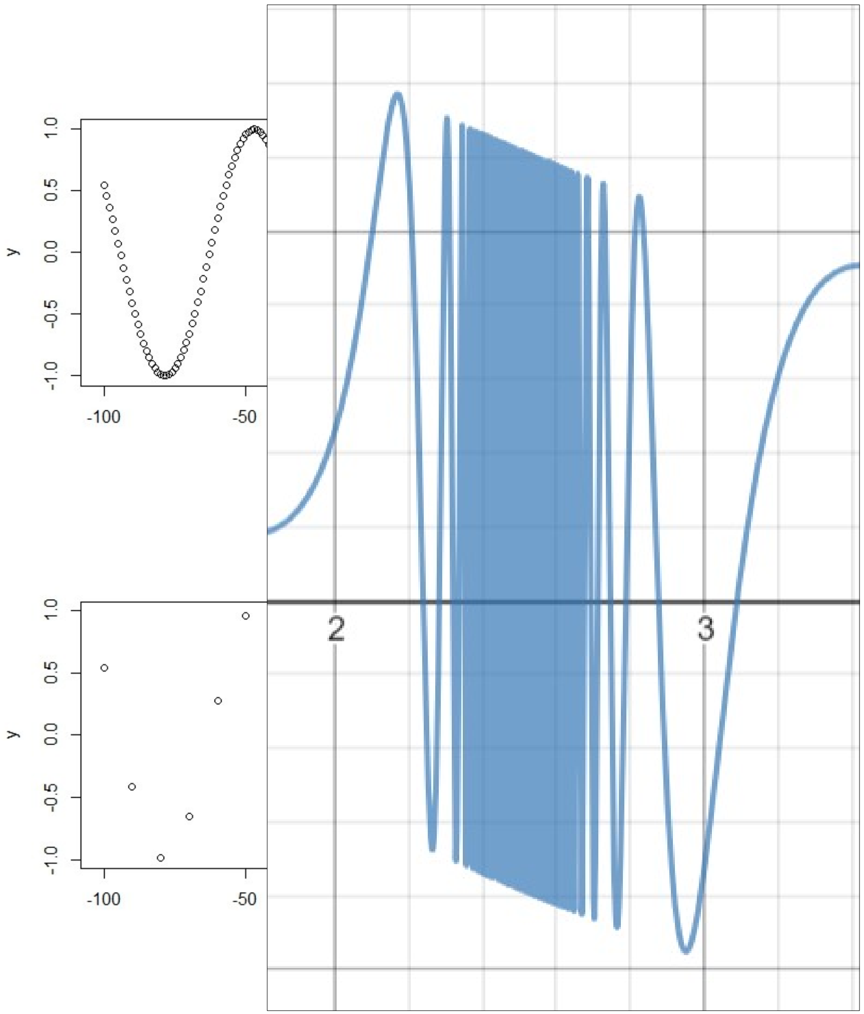
- Actual value: $\int_{-1}^{1} e^{x^2} \, \mathrm{d}x = 2.925\ldots$

Polynomial Interpolation

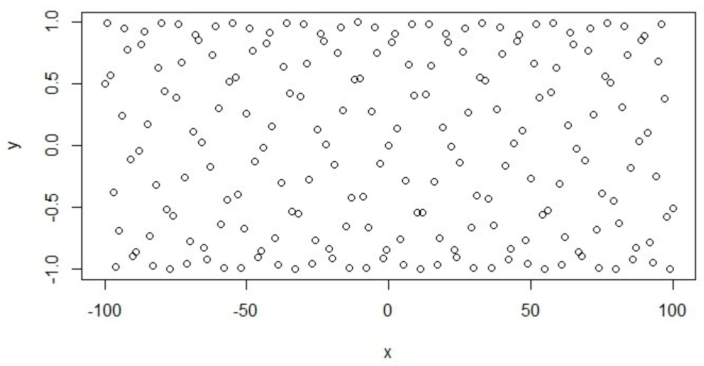


- Given a set of nodes, called *interpolation nodes*, which lie on the function to be approximated, a polynomial which intersects all the nodes will likely be a good approximation.
- The polynomial is formed by **solving a system of linear equations.**

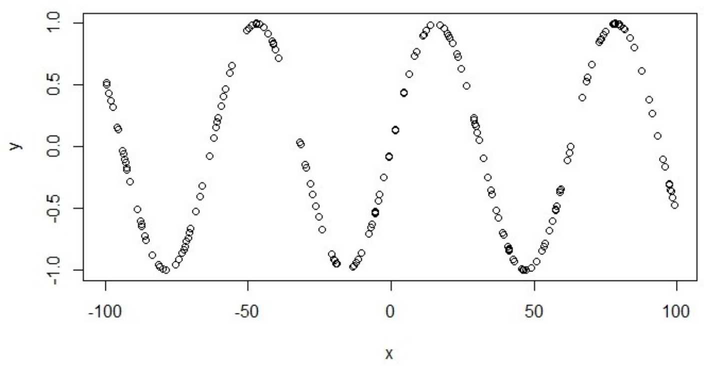
Polynomial Interpolation



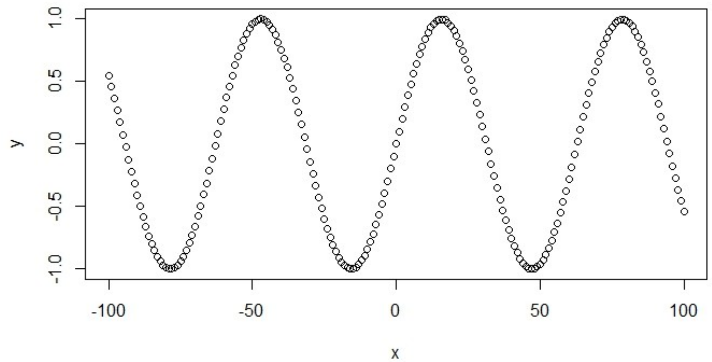
200 nodes for a highly oscillatory function



200 badly distributed nodes



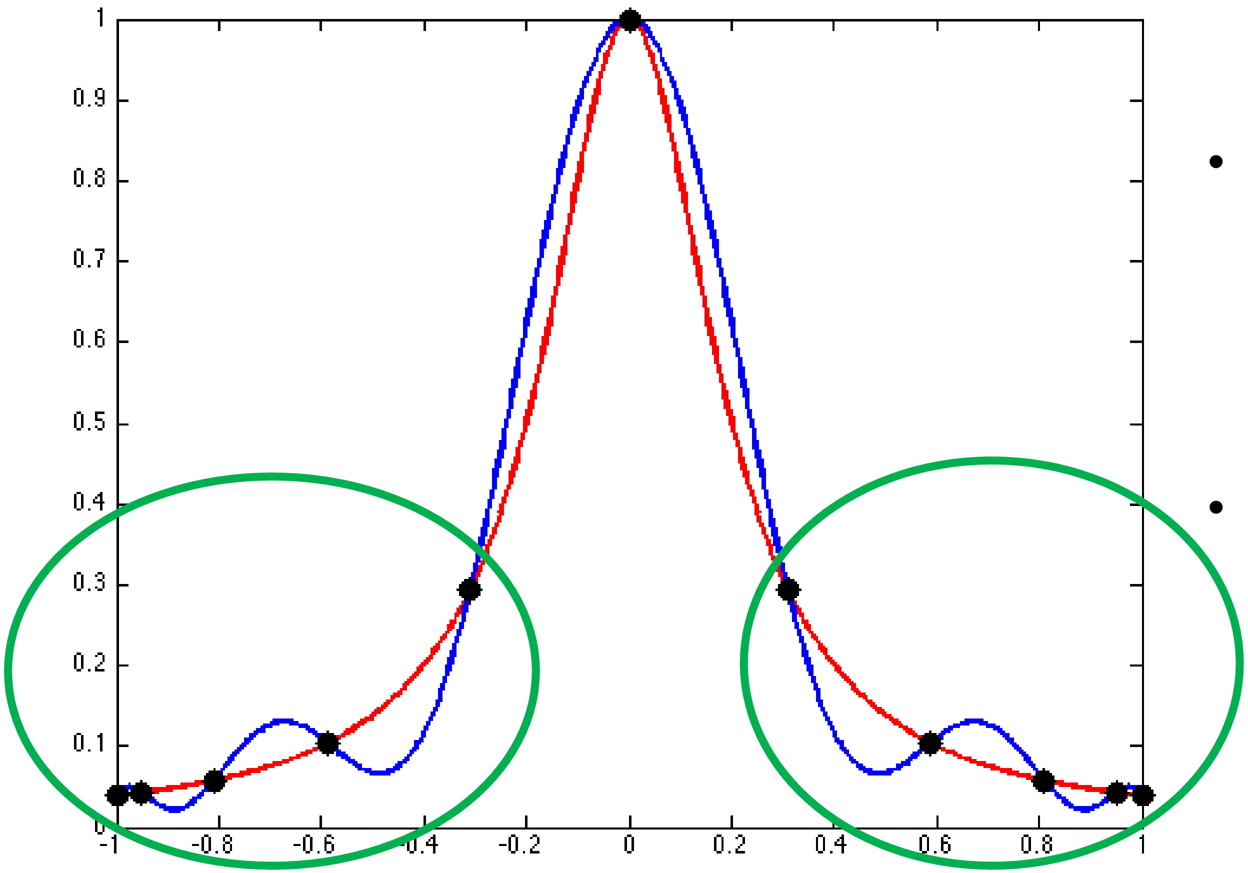
Choice of Distribution of Interpolation Nodes



200 equi-spaced nodes: x = -100, -99, -98, …, 100

- Equi-spaced nodes result in a relatively poor polynomial approximation due to the *Runge Phenomenon*.
- The use of ***Chebyshev nodes*** averts the Runge Phenomenon and results in a better approximation.

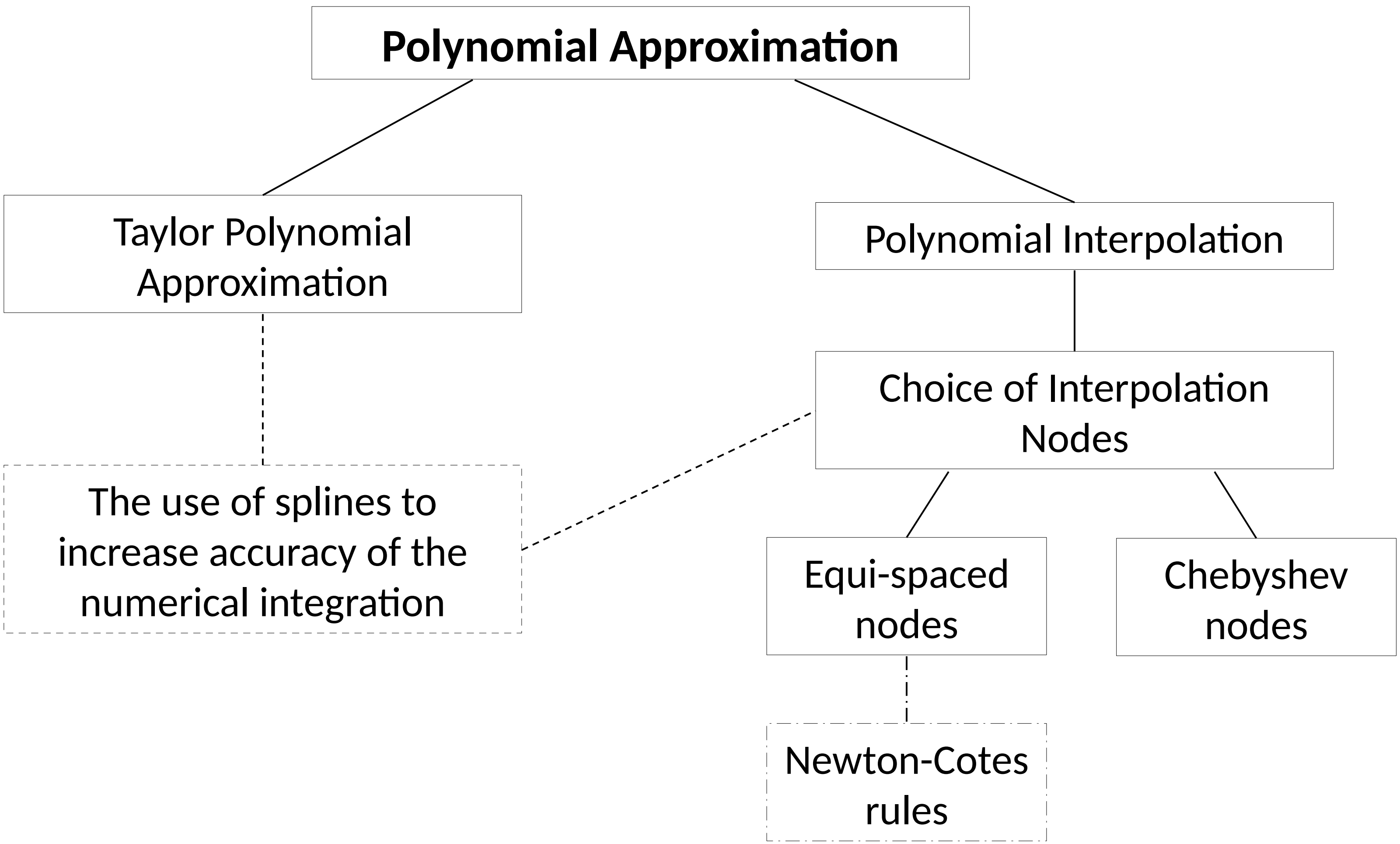
Chebyshev Nodes



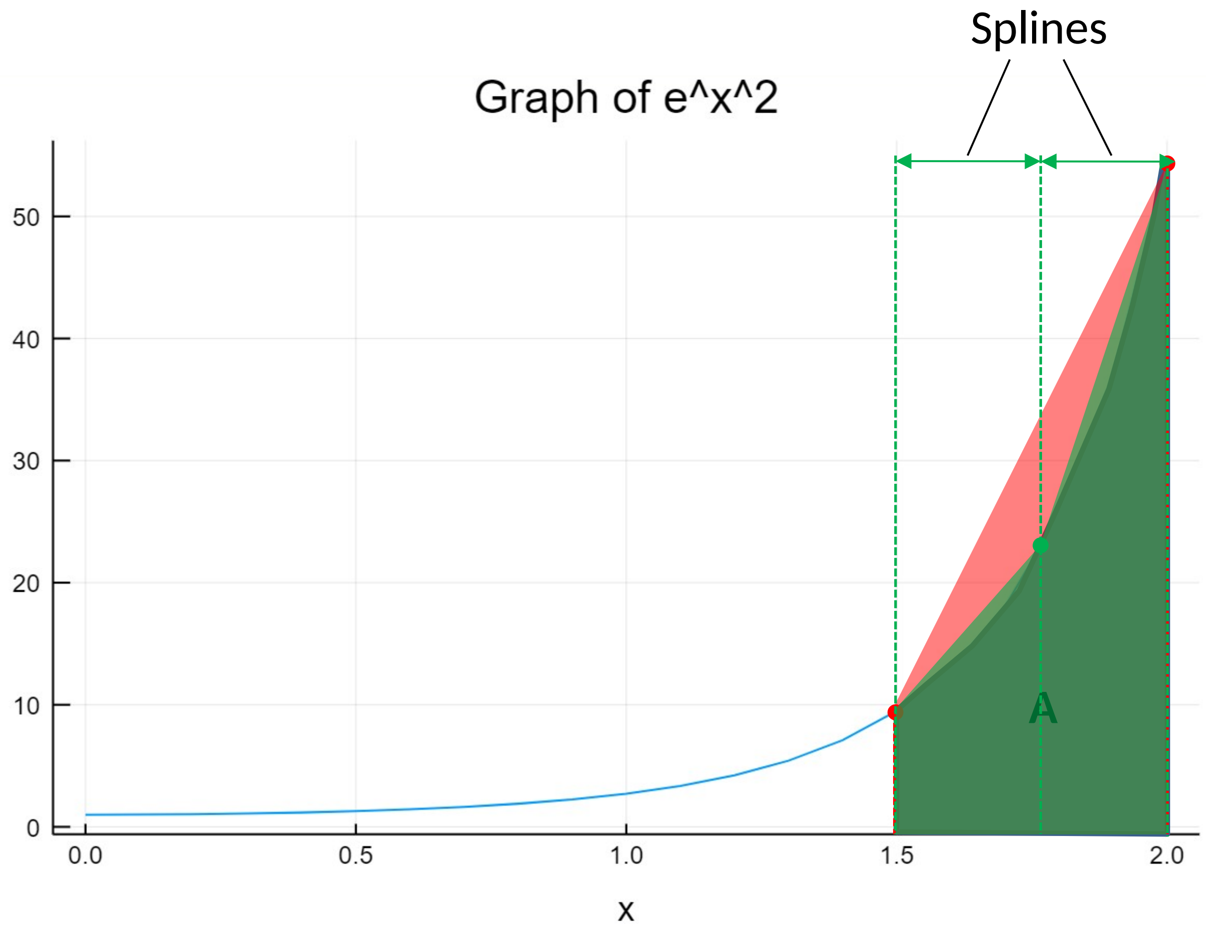
Graph taken from
https://math.boisestate.edu/~calhoun/teaching/matlab-tutorials/lab_11/html/lab_11.html

- The nodes shown are Chebyshev nodes; they cluster about the extreme ends of the graph.
- The Runge Phenomenon manifests as **oscillations at the extreme ends.**

Numerical Integration: The many considerations

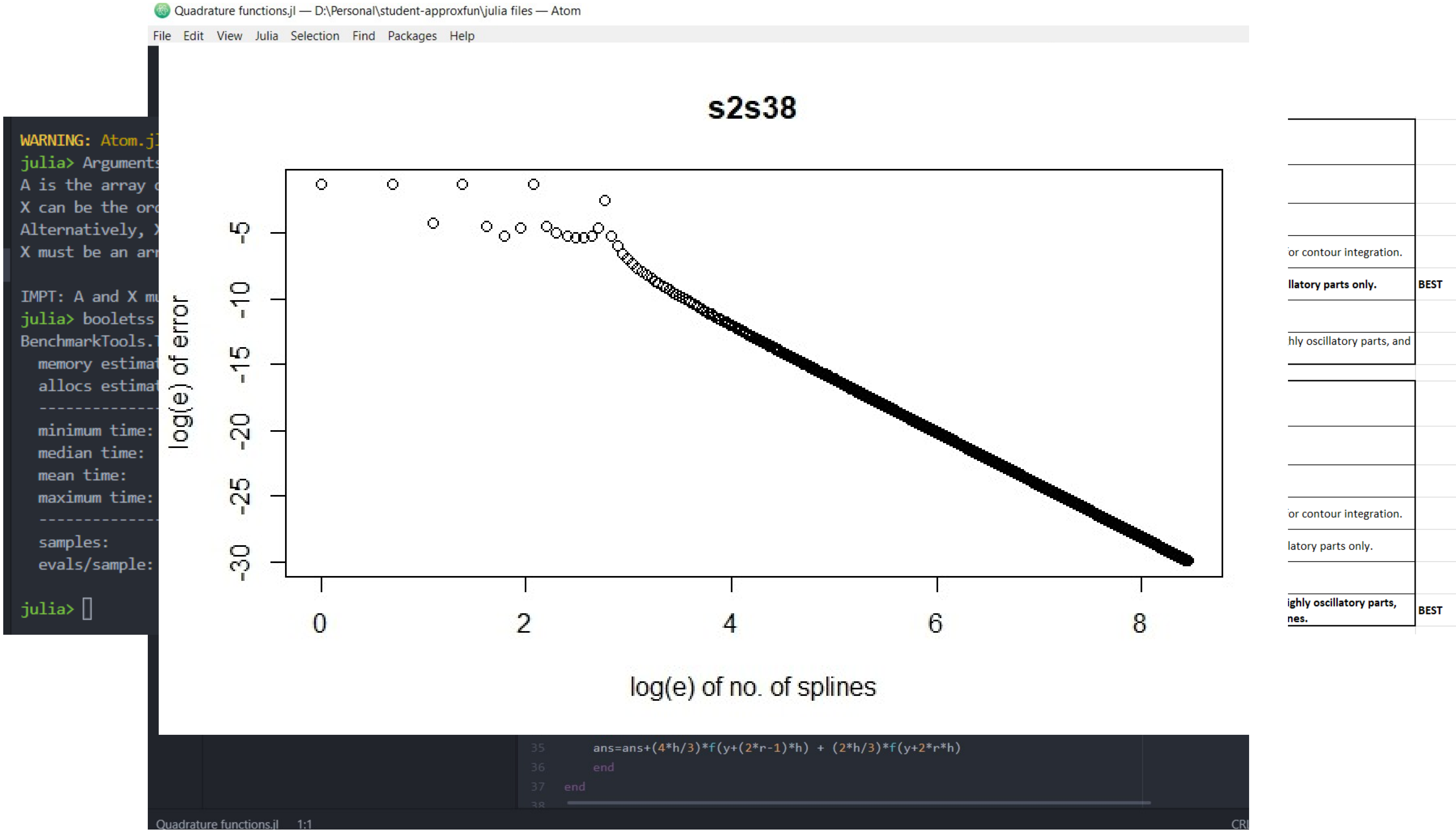


Splines and Composition



- The summation of multiple splines is called *composition*.
- The greater the number of splines used, the more accurate the numerical integration.

My Work


Quadrature functions.jl — D:\Personal\student-approxfun\julia files — Atom
File Edit View Julia Selection Find Packages Help
s2s38
log(e) of error
log(e) of no. of splines
-5
-10
-15
-20
-25
-30
0
2
4
6
8
WARNING: Atom.jl
julia> Arguments
A is the array
X can be the or
Alternatively,
X must be an ar
IMPT: A and X mu
julia> booletss
BenchmarkTools.
memory estimat
allocs estimat
minimum time:
median time:
mean time:
maximum time:
samples:
evals/sample:
julia>
ans=ans+(4*h/3)*f(y+(2*r-1)*h) + (2*h/3)*f(y+2*r*h)
end
end
Quadrature functions.jl 1:1
or contour integration.
llatory parts only.
BEST
hly oscillatory parts, and
or contour integration.
latory parts only.
ighly oscillatory parts,
nes.
BEST

Formula for the wavefunction

$$
\begin{aligned}
2\pi q(x,t) = & \int_{-\infty}^{\infty} e^{i\lambda x - i\lambda^2 t} \hat{q}_0(\lambda)\, \mathrm{d}\lambda \\
& + \int_{\delta D^+} \frac{2i\lambda}{\beta + i\lambda} (e^{i\lambda x - i\lambda^2 t}) \int_0^{\tau} e^{i\lambda^2 s} h(s)\, \mathrm{d}s\, \mathrm{d}\lambda \\
& + \int_{\delta D^+} \frac{\beta - i\lambda}{\beta + i\lambda} (e^{i\lambda x - i\lambda^2 t}) \hat{q}_0(-\lambda)\, \mathrm{d}\lambda \\
& - \int_{\delta D^+} \frac{\beta - i\lambda}{\beta + i\lambda} (e^{i\lambda x - i\lambda^2 t})(e^{i\lambda^2 \tau}) \hat{q}(-\lambda; \tau)\, \mathrm{d}\lambda
\end{aligned}
$$

The time-dependent linear Schrodinger equation simply serves as a litmus test for our algorithm!

The Algorithm

ApproxFun (Chebyshev nodes) but with some wrapper functions coded around it for increased efficiency and to gear ApproxFun to perform **numerical contour integration**.

ApproxFun.jl

docs stable | docs latest | build passing | coverage 90% | chat on gitter

ApproxFun is a package for approximating functions. It is in a similar vein to the Matlab package `Chebfun` and the Mathematica package `RHPackage`.

The `ApproxFun Documentation` contains detailed information, or read on for a brief overview of the package.

The `ApproxFun Examples` contains many examples of using this package, in Jupyter notebooks and Julia scripts.

Introduction

Take your two favourite functions on an interval and create approximations to them as simply as:

```
using LinearAlgebra, SpecialFunctions, Plots, ApproxFun
x = Fun(identity,0..10)
f = sin(x^2)
g = cos(x)
```

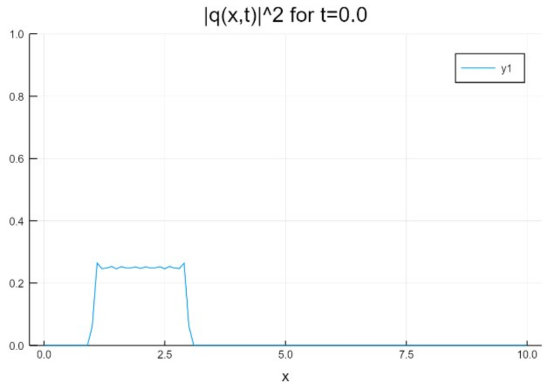
Evaluating `f(.1)` will return a high accuracy approximation to `sin(0.01)`. All the algebraic manipulations of functions are supported and more. For example, we can add `f` and `g^2` together and compute the roots and extrema:

```
h = f + g^2
r = roots(h)
rp = roots(h')
```

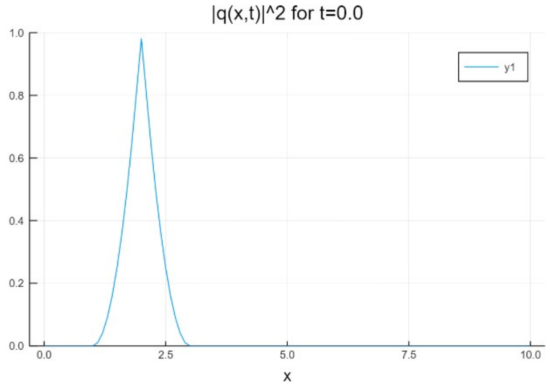
These are contours on the complex plane.

The Four Types of Initial Condition

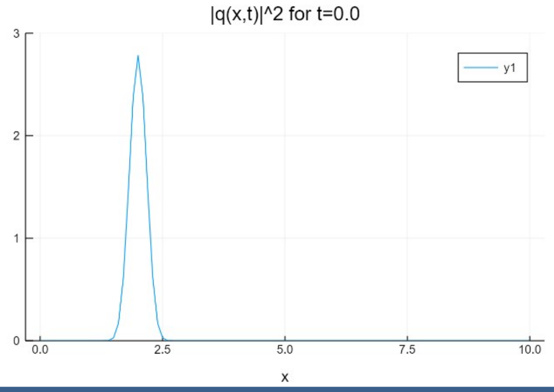
Non-continuous



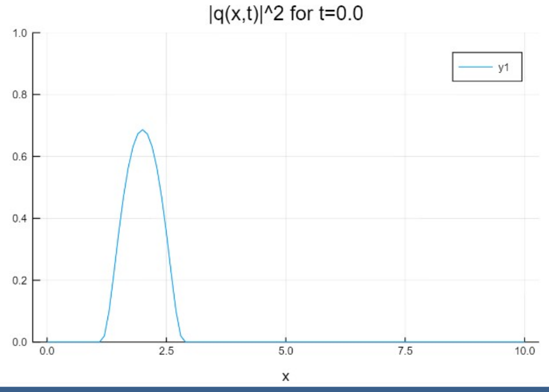
Continuous but non-differentiable



Continuous and n-differentiable

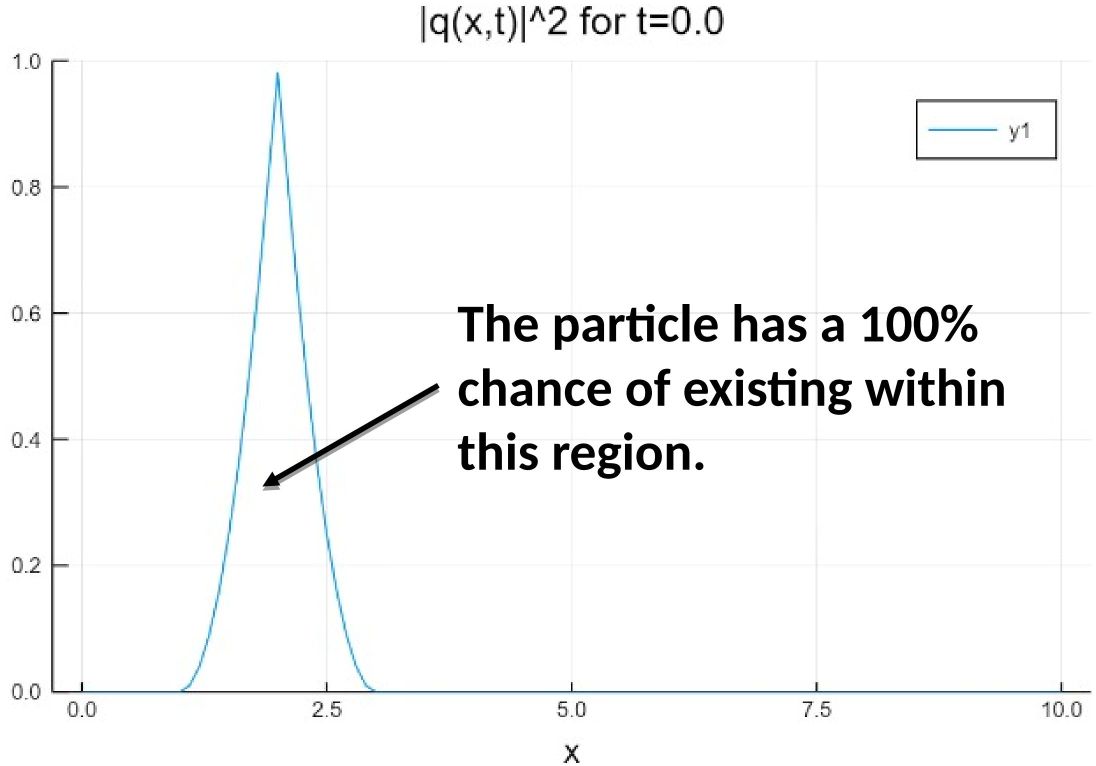


Continuous and infinitely differentiable (mollifier)



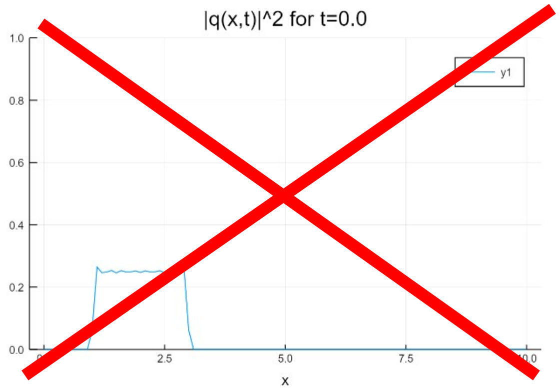
The Probability Density Function

We plotted $|q(x,t)|^2$ which is a probability density function telling us the probability of finding the quantum particle in a region of x – values.



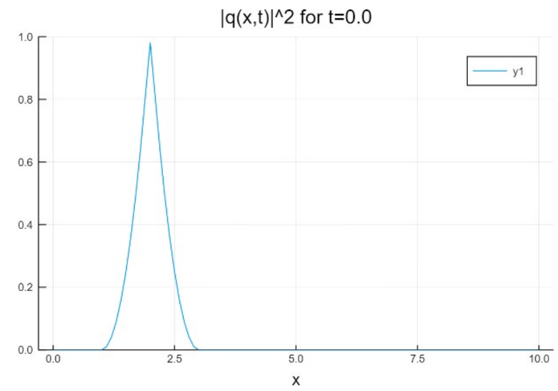
The Four Types of Initial Condition

Non-continuous

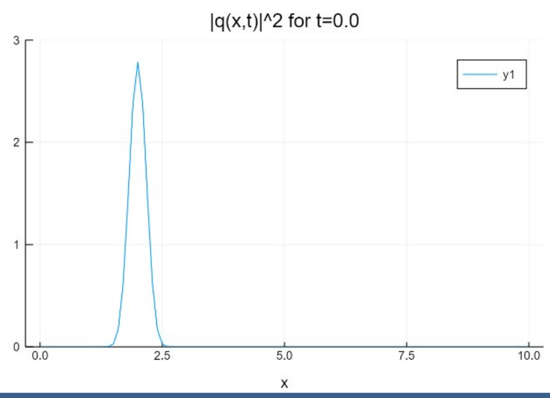


Graphs are inaccurate

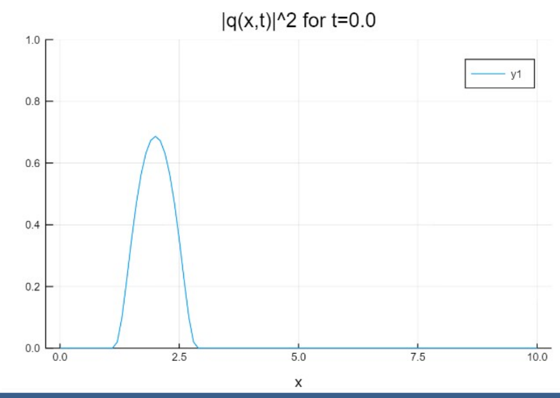
Continuous but non-differentiable

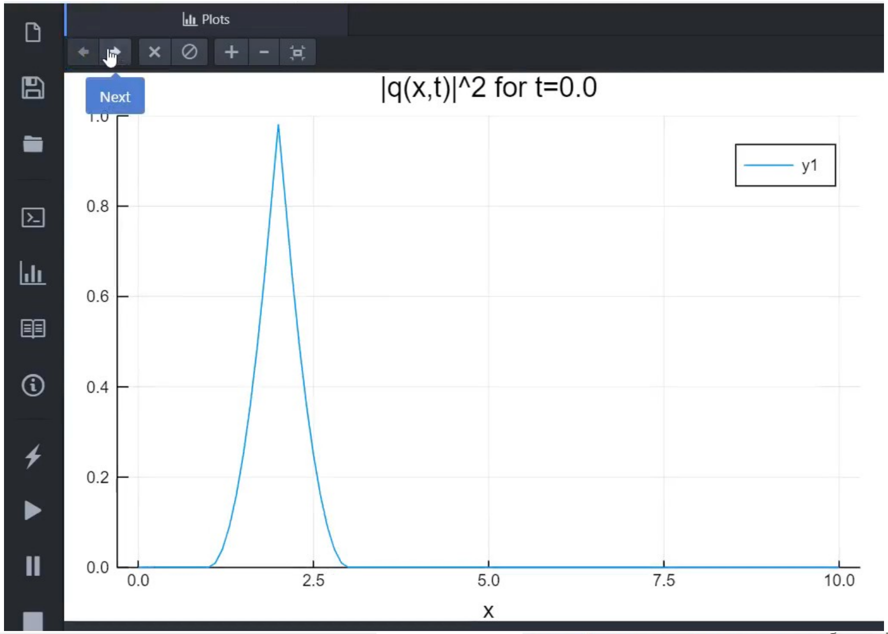


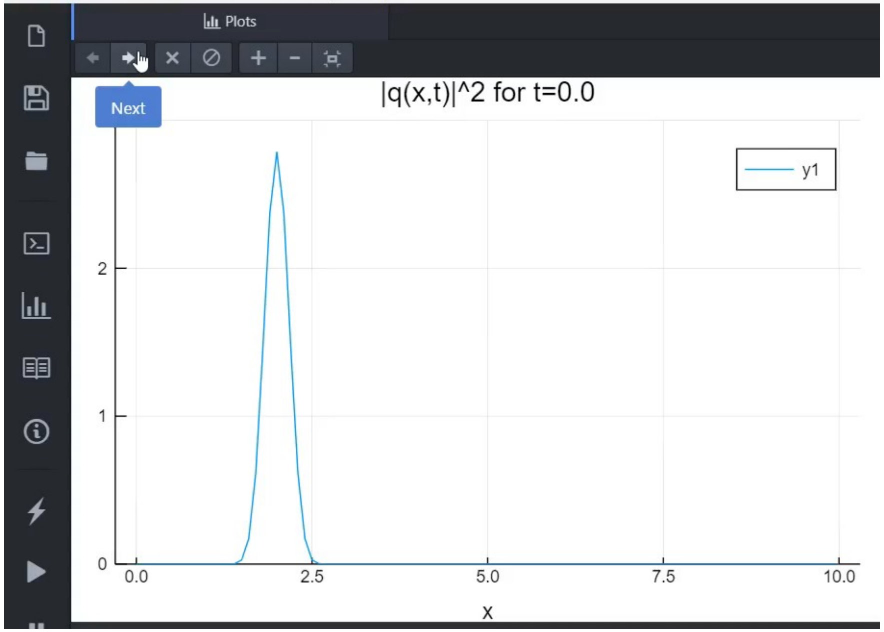
Continuous and n-differentiable

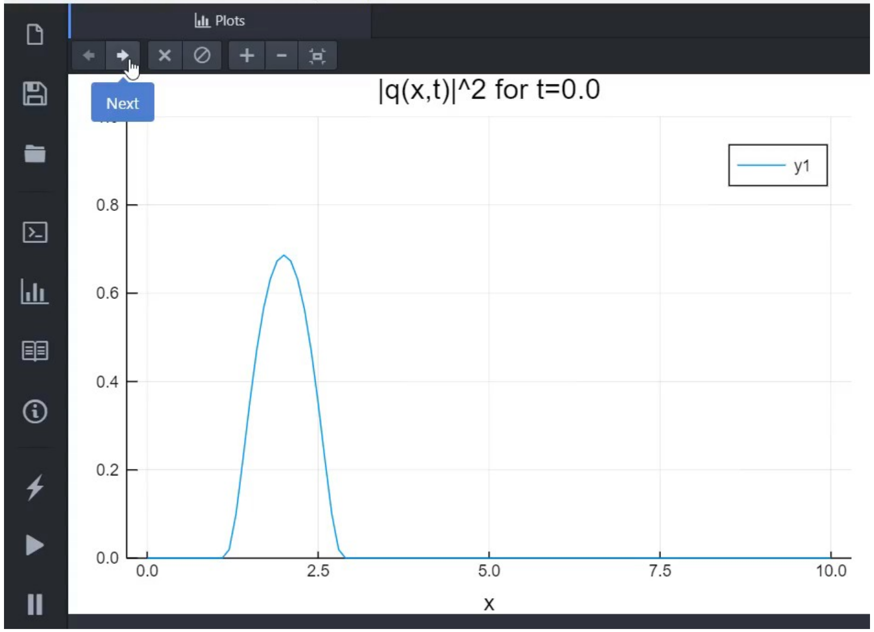


Continuous and infinitely differentiable (mollifier)





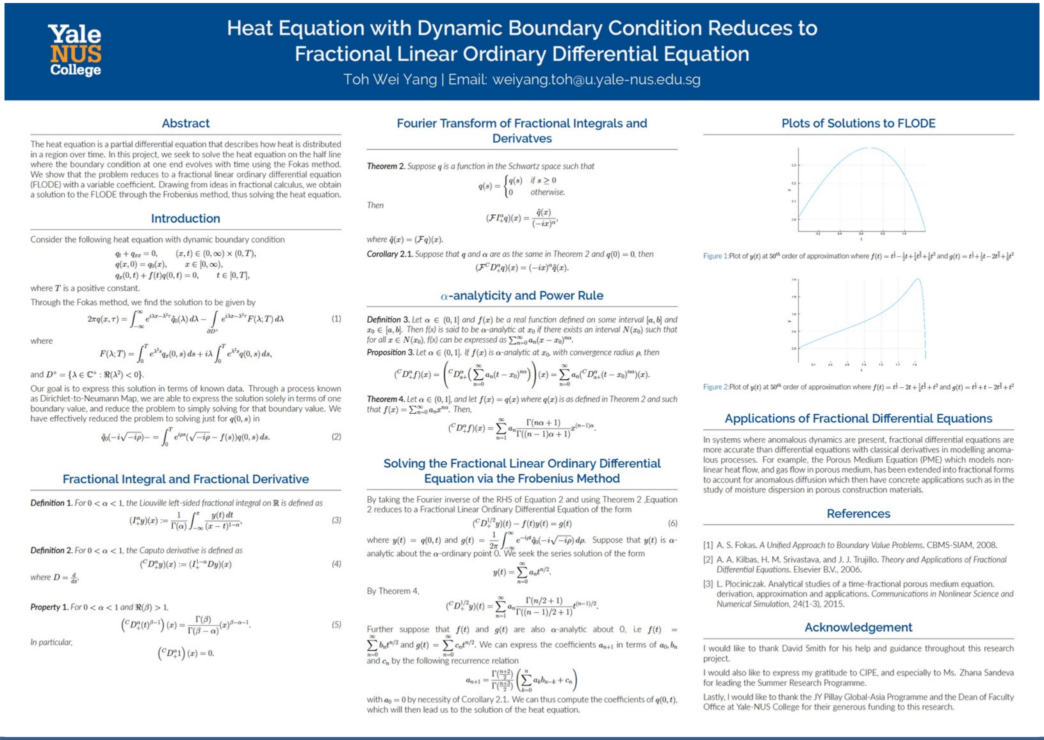




YaleNUSCollege

Other Potential Applications of this Algorithm

- Many problems require numerical integration, but the numerical integration algorithm best suited to each problem may be different.
- This algorithm can be used to help solve problems on heat flow. Toh Wei Yang's project dealt with the heat equation, which models heat flow:



Acknowledgements

- Assistant Professor David Andrew Smith, Yale-NUS College

- JY Pillay Global-Asia Programme
- Dean of Faculty Office, Yale-NUS College
- Centre for International and Professional Experience, Yale-NUS College, especially Ms Zhana Sandeva.

References

References

[1] M. J. Ablowitz and A. S. Fokas, *Complex variables*, Cambridge Texts in Applied Mathematics, Cambridge University Press, 1997.

[2] A. S. Fokas, *A unified approach to boundary value problems*, CBMS-SIAM, 2008.

[3] Wouter Den Haan, http://econ.lse.ac.uk/staff/wdenhaan/numerical/, filename: functionapproximation.pdf, last modified: 2011-08-29 16:02.

[4] J.H. Mathews and K.D. Fink, *Numerical methods using matlab*, Featured Titles for Numerical Analysis Series.

[5] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery, *Numerical recipes in c (2nd ed.): The art of scientific computing*, pp. 190–195, Cambridge University Press, New York, NY, USA, 1992.

[6] L.N. Trefethen, *Finite difference and spectral methods for ordinary and partial differential equations*, pp. 260–268, The author, 1996.