Pácticas de Análisis Matemático con Julia

Alfredo Sánchez Alberca
asalber@ceu.es
https://aprendeconalf.es

Indice de contenidos

Prefacio

¡Bienvenido a Prácticas de Análisis Matemático con Julia!

Este libro presenta una recopilación de prácticas de Análisis Matemático en una y varias variables reales con el lenguaje de programación Julia, con problemas aplicados a las Ciencias y las Ingenierías.

No es un libro para aprender a programar con Julia, ya que solo enseña el uso del lenguaje y de algunos de sus paquetes para resolver problemas de Cálculo tanto numérico como simbólico. Para quienes estén interesados en aprender a programar en este Julia, os recomiendo leer este manual de Julia.

Licencia

Al reutilizar o distribuir la obra, tiene que dejar bien claro los términos de la licencia de esta obra.

Estas condiciones pueden no aplicarse si se obtiene el permiso del titular de los derechos de autor.

Nada en esta licencia menoscaba o restringe los derechos morales del autor.

1 Introducción

La gran potencia de cálculo alcanzada por los ordenadores en las últimas décadas ha convertido a los mismos en poderosas herramientas al servicio de todas aquellas disciplinas que, como las matemáticas, requieren cálculos largos y complejos.

Julia es un lenguaje de programación especialmente orientado al cálculo numérico y el análisis de datos. Julia permite además realizar cálculos simbólicos y dispone de una gran biblioteca de paquetes con aplicaciones en muy diversas áreas de las Matemáticas como Cálculo, Álgebra, Geometría, Matemática Discreta o Estadística.

La ventaja de Julia frente a otros programas habituales de cálculo como Mathematica, MATLAB o Sage radica en su potencia de cálculo y su velocidad (equiparable al lenguaje C), lo que lo hace ideal para manejar grandes volúmenes de datos o realizar tareas que requieran largos y complejos cálculos. Además, es software libre por lo que resulta ideal para introducirlo en el aula como soporte computacional para los modelos matemáticos sin coste alguno.

En el siguiente enlace se explica el procedimiento de instalación de Julia.

Existen también varios entornos de desarrollo online que permiten ejecutar código en Julia sin necesidad de instalarlo en nuestro ordenador, como por ejemplo Replit, Cocalc o Codeanywhere.

El objetivo de esta práctica es introducir al alumno en la utilización de este lenguaje, enseñándole a realizar las operaciones básicas más habituales en Cálculo.

1.1 El REPL de Julia

Para arrancar el REPL^(REPL es el acrónimo de Read, Evaluate, Print and Loop, que describe el funcionamiento del compilador de Julia) de julia basta con abrir una terminal y teclear `julia`.

```
prompt> julia
               _
   _       _ _(_)_     |  Documentation: https://docs.julialang.org
  (_)     | (_) (_)    |
   _ _   _| |_  __ _   |  Type "?" for help, "]?" for Pkg help.
  | | | | | | |/ _` |  |
  | | |_| | | | (_| |  |  Version 1.7.3 (2022-05-06)
 _/ |\__'_|_|_|\__'_|  |  Official https://julialang.org/ release
|__/                   |

julia>
```

1.2 El gestor de paquetes de Julia

Julia viene con varios paquetes básicos preinstalados, como por ejemplo el paquete `LinearAlgebra` que define funciones básicas del Álgebra Lineal, pero en estas prácticas utilizaremos otros muchos paquetes que añaden más funcionalidades que no vienen instalados por defecto y tendremos que instalarlos aparte. Julia tiene un potente gestor de paquetes que facilita la búsqueda, instalación, actualización y eliminación de paquetes.

Por defecto el gestor de paquetes utiliza el repositorio de paquetes oficial pero se pueden instalar paquetes de otros repositorios.

Para entrar en el modo de gestión de paquetes hay que teclear `]`. Esto produce un cambio en el *prompt* del REPL de Julia.

Los comandos más habituales son:

- `add p`: Instala el paquete `p` en el entorno activo de Julia.
- `update`: Actualiza los paquetes del entorno activo de Julia.
- `status`: Muestra los paquetes instalados y sus versiones en el entorno activo de Julia.

- `remove p`: Elimina el paquete `p` del entorno activo de Julia.

> **Ejemplo**
>
> Para instalar el paquete `SymPy` para cálculo simbólico basta con teclear `add Sympy`.

```
(@v1.7) pkg> add SymPy
    Updating registry at `~/.julia/registries/General.toml`
   Resolving package versions...
    Updating `~/.julia/environments/v1.7/Project.toml`
  [24249f21] + SymPy v1.1.6
    Updating `~/.julia/environments/v1.7/Manifest.toml`
  [3709ef60] + CommonEq v0.2.0
  [38540f10] + CommonSolve v0.2.1
  [438e738f] + PyCall v1.93.1
  [24249f21] + SymPy v1.1.6
```

1.3 Operadores aritméticos.

El uso más simple de Julia es la realización de operaciones aritméticas como en una calculadora. En Julia se utilizan los siguientes operadores.

Operador	Descripción
`x + y`	Suma
`x - y`	Resta
`x * y`	Producto
`x / y`	División
`x ÷ y`	Cociente división entera
`x % y`	Resto división entera
`x ^ y`	Potencia

1.4 Operadores de comparación

Operador	Descripción
`==`	Igualdad
`!=`,	Desigualdad
`<`	Menor que
`<=`,	Menor o igual que

Operador	Descripción
`>`	Mayor que
`>=`,	Mayor o igual que

1.5 Operadores booleanos

Operador	Descripción
`!x`	Negación
`x && y`	Conjunción (y)
`x || y`	Disyunción (o)

Existen también un montón de funciones predefinidas habituales en Cálculo.

1.6 Funciones de redondeo

Función	Descripción
`round(x)`	Devuelve el entero más próximo a `x`
`round(x, digits = n)`	Devuelve al valor más próximo a `x` con `n` decimales
`floor(x)`	Redondea `x` al próximo entero menor
`ceil(x)`	Redondea `x` al próximo entero mayor
`trunc(x)`	Devuelve la parte entera de `x`

Ejemplo

```
julia> round(2.7)
3.0

julia> floor(2.7)
2.0

julia> floor(-2.7)
-3.0

julia> ceil(2.7)
3.0

julia> ceil(-2.7)
-2.0

julia> trunc(2.7)
2.0

julia> trunc(-2.7)
-2.0

julia> round(2.5)
2.0

julia> round(2.786, digits = 2)
2.79
```

1.7 Funciones de división

Función	Descripción
`div(x,y)`, `x÷y`	Cociente de la división entera
`fld(x,y)`	Cociente de la división entera redondeado hacia abajo
`cld(x,y)`	Cociente de la división entera redondeado hacia arriba
`rem(x,y)`, `x%y`	Resto de la división entera. Se cumple `x == div(x,y)*y + rem(x,y)`
`mod(x,y)`	Módulo con respecto a `y`. Se cumple `x == fld(x,y)*y + mod(x,y)`
`gcd(x,y...)`	Máximo común divisor positivo de `x`, `y`,...

Función	Descripción
`lcm(x,y...)`	Mínimo común múltiplo positivo de `x`, `y`,...

Ejemplo

```
julia> div(5,3)
1

julia> cld(5,3)
2

julia> 5%3
2

julia> -5%3
-2

julia> mod(5,3)
2

julia> mod(-5,3)
1

julia> gcd(12,18)
6

julia> lcm(12,18)
36
```

1.8 Funciones para el signo y el valor absoluto

Función	Descripción
`abs(x)`	Valor absoluto de `x`
`sign(x)`	Devuelve -1 si `x` es positivo, -1 si es negativo y 0 si es 0.

Ejemplo

```
julia> abs(2.5)
2.5

julia> abs(-2.5)
2.5

julia> sign(-2.5)
-1.0

julia> sign(0)
0

julia> sign(2.5)
1.0
```

1.9 Raíces, exponenciales y logaritmos

Función	Descripción
`sqrt(x)`, √x	Raíz cuadrada de `x`
`cbrt(x)`, ∛x	Raíz cúbica de `x`
`exp(x)`	Exponencial de `x`
`log(x)`	Logaritmo neperiano de `x`
`log(b,x)`	Logaritmo en base `b` de `x`
`log2(x)`	Logaritmo en base 2 de `x`
`log10(x)`	Logaritmo en base 10 de `x`

Ejemplo

```
julia> sqrt(4)
2.0

julia> cbrt(27)
3.0

julia> exp(1)
2.718281828459045

julia> exp(-Inf)
0.0

julia> log(1)
0.0

julia> log(0)
-Inf

julia> log(-1)
ERROR: DomainError with -1.0:
log will only return a complex result if called with a complex argument.
...

julia> log(-1+0im)
0.0 + 3.141592653589793im

julia> log2(2^3)
3.0
```

1.10 Funciones trigonométricas

Función	Descripción
`hypot(x,y)`	Hipotenusa del triángulo rectángulo con catetos `x` e `y`
`sin(x)`	Seno del ángulo `x` en radianes
`sind(x)`	Seno del ángulo `x` en grados
`cos(x)`	Coseno del ángulo `x` en radianes
`cosd(x)`	Coseno del ángulo `x` en grados

Función	Descripción
`tan(x)`	Tangente del ángulo `x` en radianes
`tand(x)`	Tangente del ángulo `x` en grados
`sec(x)`	Secante del ángulo `x` en radianes
`csc(x)`	Cosecante del ángulo `x` en radianes
`cot(x)`	Cotangente del ángulo `x` en radianes

Ejemplo

```
julia> sin( /2)
1.0

julia> cos( /2)
6.123233995736766e-17

julia> cosd(90)
0.0

julia> tan( /4)
0.9999999999999999

julia> tand(45)
1.0

julia> tan( /2)
1.633123935319537e16

julia> tand(90)
Inf

julia> sin( /4)^2 + cos( /4)^2
1.0
```

1.11 Funciones trigonométricas inversas

Función	Descripción
`asin(x)`	Arcoseno (inversa del seno) de `x` en radianes
`asind(x)`	Arcoseno (inversa del seno) de `x` en grados

Función	Descripción
`acos(x)`	Arcocoseno (inversa del coseno) de `x` en radianes
`acosd(x)`	Arcocoseno (inversa del coseno) de `x` en grados
`atan(x)`	Arcotangente (inversa de la tangente) de `x` en radianes
`atand(x)`	Arcotangente (inversa de la tangente) de `x` en grados
`asec(x)`	Arcosecante (inversa de la secante) de `x` en radianes
`acsc(x)`	Arcocosecante (inversa de la cosecante) de `x` en radianes
`acot(x)`	Arcocotangente (inversa de la cotangente) de `x` en radianes

i Ejemplo

```
julia> asin(1)
1.5707963267948966

julia> asind(1)
90.0

julia> acos(-1)
3.141592653589793

julia> atan(1)
0.7853981633974483

julia> atand(tan( /4))
45.0
```

1.12 Precedencia de operadores

A la hora de evaluar una expresión aritmética, Julia evalúa los operadores según el siguiente orden de prioridad (de mayor a menor prioridad).

Categoría	Operadores	Asociatividad
Funciones	`exp`, `log`, `sin`, etc.	
Exponenciación	`^`	Derecha
Unarios	`+ - √`	Derecha
Fracciones	`//`	Izquierda
Multiplicación	`* / % & \ ÷`	Izquierda
Adición	`+ - |`	Izquierda
Comparaciones	`> < >= <= == != !==`	

Categoría	Operadores	Asociatividad
Asignaciones	`= += -= *= /= //= ^= ÷= %= |= &=`	Derecha

Cuando se quiera evaluar un operador con menor prioridad antes que otro con mayor prioridad, hay que utilizar paréntesis.

Ejemplo

```
julia> 1 + 4 ^ 2 / 2 - 3
6.0

julia> (1 + 4 ^ 2) / 2 - 3
5.5

julia> (1 + 4) ^ 2 / 2 - 3
9.5

julia> 1 + 4 ^ 2 / (2 - 3)
-15.0

julia> (1 + 4 ^ 2) / (2 - 3)
-17.0
```

1.13 Definición de variables

Para definir variables se pueden utilizar cualquier carácter Unicode. Los nombres de las variables pueden contener más de una letra y, en tal caso, pueden usarse también números, pero siempre debe comenzar por una letra. Así, para Julia, la expresión `xy`, no se interpreta como el producto de la variable x por la variable y, sino como la variable xy. Además, se distingue entre mayúsculas y minúsculas, así que no es lo mismo xy que xY.

2 Sucesiones de números reales

2.1 Ejercicios Resueltos

Para la realización de esta práctica se requieren los siguientes paquetes:

```
using SymPy  # Para el cálculo simbólico de límites.
using Plots  # Para el dibujo de gráficas.
using LaTeXStrings  # Para usar código LaTeX en los gráficos.
```

Ejercicio 2.1. Dar los 10 primeros términos de las siguientes sucesiones:

a. $(2n+1)_{n=1}^{\infty}$

Pista

Definir una función para el término general y aplicar la función a los naturales de 1 a 10 usando compresiones de arrays.

Solución

```
x(n) = 2n + 1
print([x(n) for n = 1:10])
```

```
[3, 5, 7, 9, 11, 13, 15, 17, 19, 21]
```

b. $\left(\frac{1}{n}\right)_{n=1}^{\infty}$

Solución

```
# Como reales
x(n) = 1 / n
print([x(n) for n = 1:10])
# Como racionales
x(n) = 1//n
print([x(n) for n = 1:10])
```

```
[1.0, 0.5, 0.3333333333333333, 0.25, 0.2, 0.16666666666666666, 0.14285714285714285, 0.1
```

c. $((-1)^n)_{n=1}^{\infty}$

Solución

```
x(n) = (-1)^n
print([x(n) for n = 1:10])
```

```
[-1, 1, -1, 1, -1, 1, -1, 1, -1, 1]
```

d. $\left(\left(1+\frac{1}{n}\right)^n\right)_{n=1}^{\infty}$

Solución

```
x(n) = (1 + 1 / n)^n
print([x(n) for n = 1:10])
```

```
[2.0, 2.25, 2.3703703703703702, 2.44140625, 2.4883199999999994, 2.5216263717421135, 2.5
```

d. $x_1 = 1$ y $x_{n+1} = \sqrt{1+x_n}\ \forall n \in \mathbb{N}$

Solución

```
x(n) = n == 1 ? 1 : sqrt(1+x(n-1))
print([x(n) for n = 1:10])
```

```
Real[1, 1.4142135623730951, 1.5537739740300374, 1.5980531824786175, 1.6118477541252516,
```

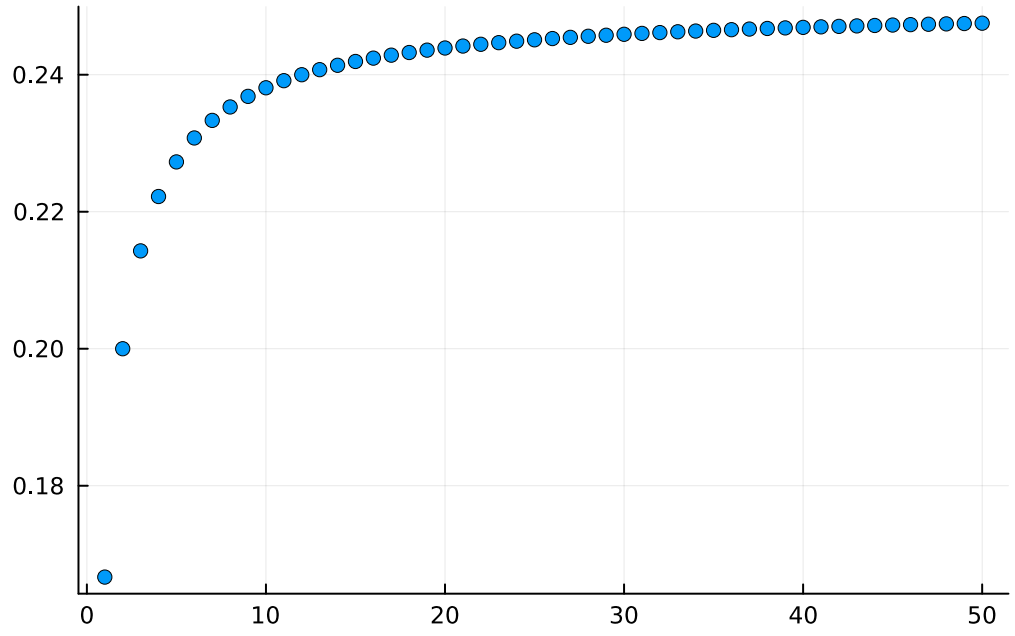
Ejercicio 2.2. Dibujar en una gráfica los 50 primeros términos de las siguientes sucesiones y deducir si son convergentes o no. En el caso de que sean convergentes, dar un valor aproximado de su límite.

> **Pista**
>
> Definir una función para el término general y aplicar la función a los naturales de 1 a 50 usando compresiones de arrays como en el ejercicio anterior. Después usar la función `scatter` del paquete `Plots` para dibujar el array de términos.

a. $\left(\frac{n}{4n+2}\right)_{n=1}^{\infty}$

> **Solución**

```
using Plots
x(n) = n / (4n + 2)
scatter([x(n) for n = 1:50], legend=false)
```

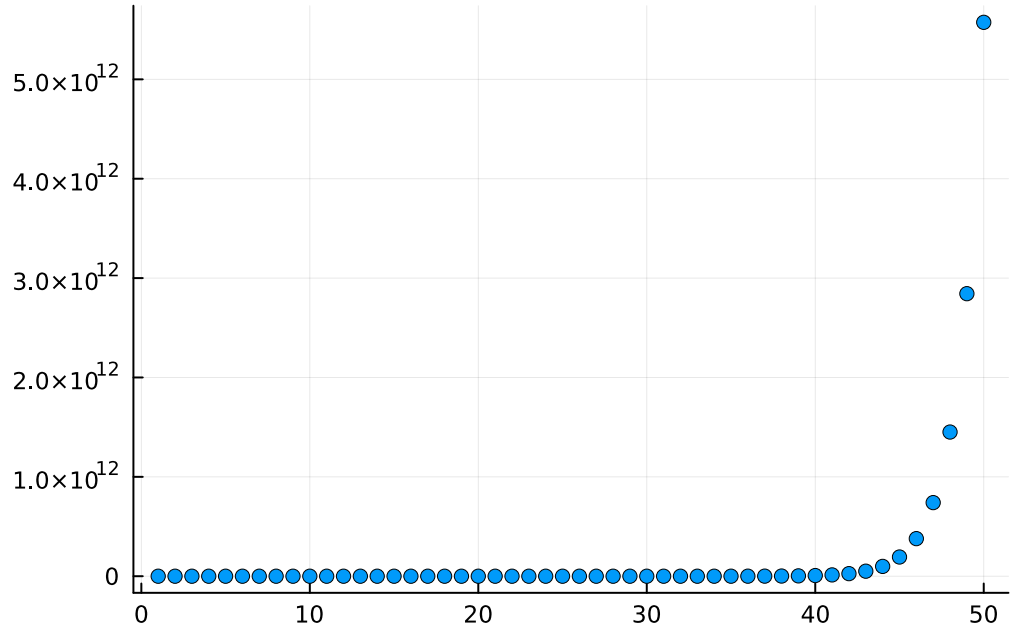


La sucesión converge al número 0.25.

a. $\left(\frac{2^n}{n^2}\right)_{n=1}^{\infty}$

Solución

```
using Plots
x(n) = 2^n / (4n + 2)
scatter([x(n) for n = 1:50], legend=false)
```

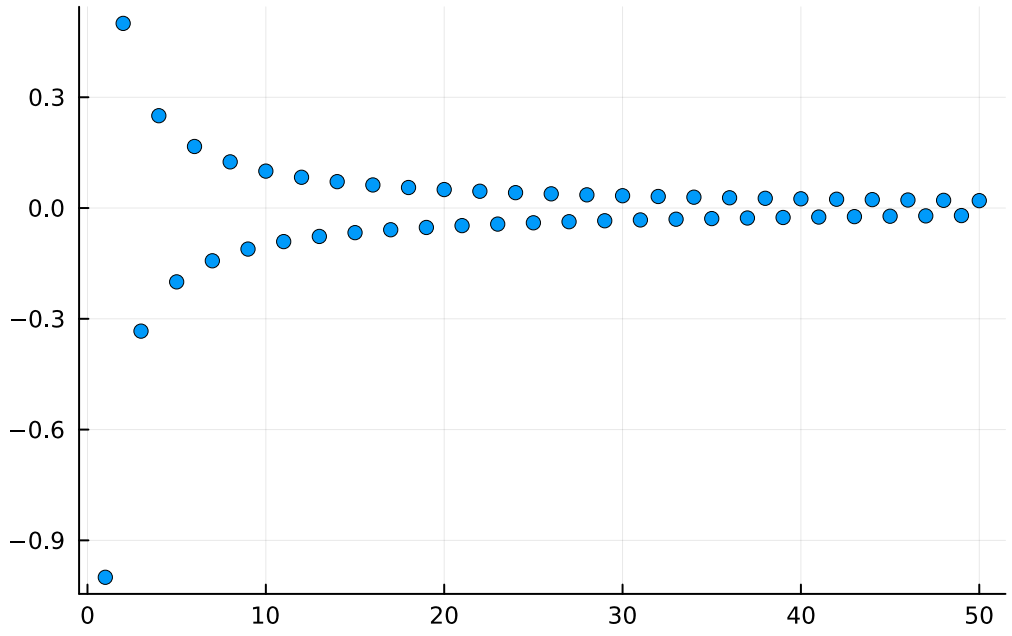


La sucesión diverge.

a. $\left(\frac{(-1)^n}{n}\right)_{n=1}^{\infty}$

Solución

```
using Plots
x(n) = (-1)^n / n
scatter([x(n) for n = 1:50], legend=false)
```

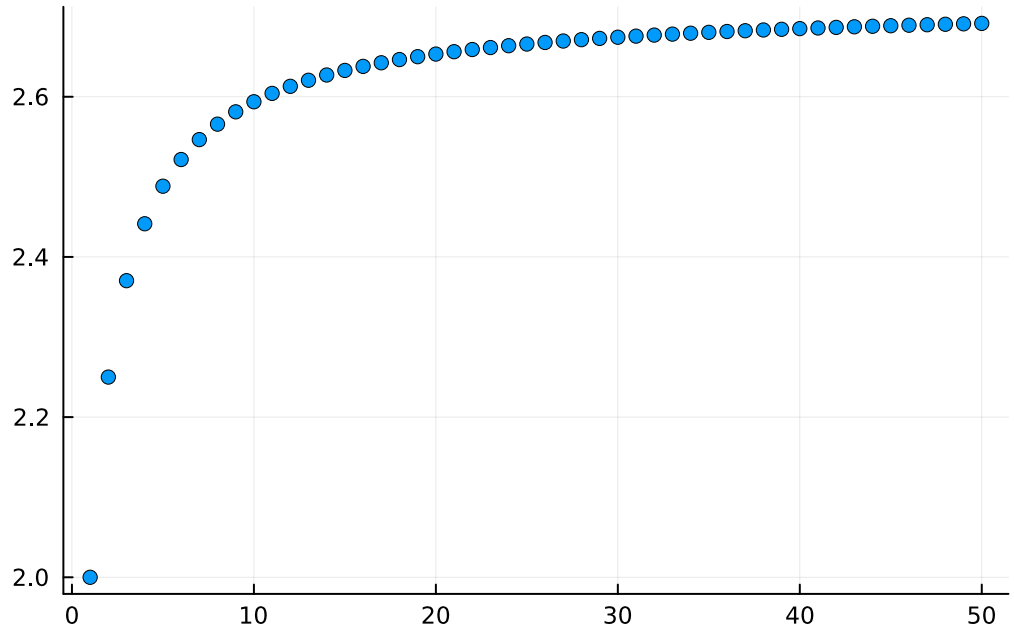


La sucesión converge al 0.

a. $\left(\left(1+\frac{1}{n}\right)^n\right)_{n=1}^{\infty}$

Solución

```
using Plots
x(n) = (1 + 1 / n)^n
scatter([x(n) for n = 1:50], legend=false)
```

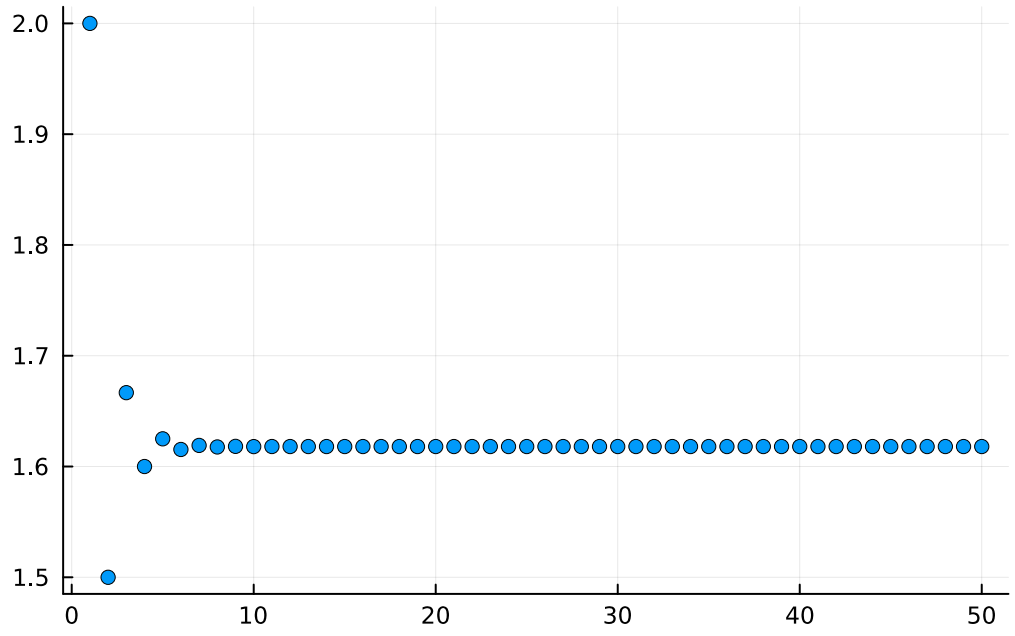


La sucesión converge aproximadamente a 2.7.

a. $x_1 = 2$ y $x_{n+1} = 1 + \frac{1}{x_n}$ $\forall n \in \mathbb{N}$

Solución

```
using Plots
x(n) =  n == 1 ? 2 : 1 + 1 / x(n-1)
scatter([x(n) for n = 1:50], legend=false)
```



La sucesión converge aproximadamente a 1.62.

Ejercicio 2.3. Calcular el límite, si existe, de las siguiente sucesiones.

a. $\left(\frac{1}{n}\right)_{n=1}^{\infty}$

Pista

Definir una función para el término general usar la función `limit` del paquete `SymPy` para calcular el límite de la sucesión.

Solución

```
using SymPy
@syms n::(integer, positive)  # Declaración de la variable simbólica n.
x(n) = 1/n
limit(x(n), n=>oo)
```

0

c. $((-1)^n)_{n=1}^{\infty}$

Solución

```
@syms n::(integer, positive)
x(n) = (-1)^n
limit(x(n), n=>oo)
```

NaN

d. $\left(\left(1+\frac{1}{n}\right)^n\right)_{n=1}^{\infty}$

Solución

```
@syms n::(integer, positive)
x(n) = (1 + 1 / n)^n
limit(x(n), n=>oo)
```

e

Ejercicio 2.4. En el siglo III A.C Arquímedes usó el método por agotamiento para calcular el área encerrada por una circunferencia (y de paso el valor de π). La idea consiste en inscribir la circunferencia en polígonos regulares con un número de lados cada vez mayor.

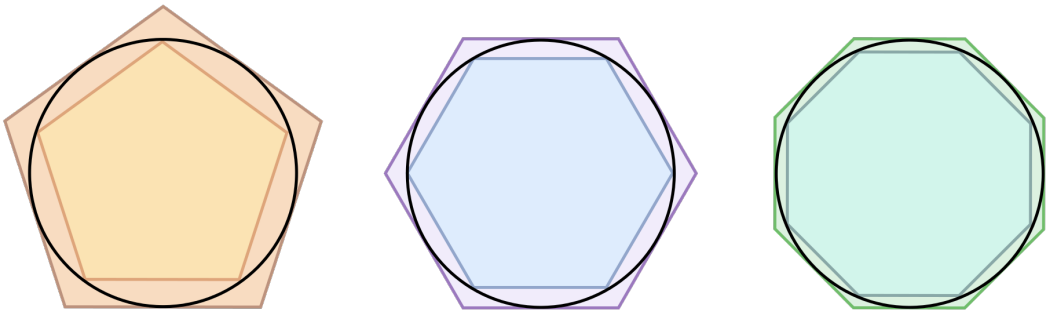


Figura 2.1: Aproximación del área de una circunferencia mediante polígonos regulares

El área de estos polígonos puede calcularse fácilmente descomponiendo los polígonos regulares en triángulos como en el siguiente ejemplo.

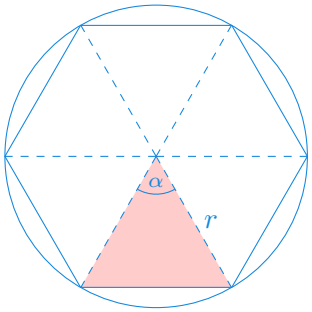


Figura 2.2: Descomposición de un hexágono en triángulos

En el caso de los polígonos inscritos dentro de la circunferencia, como dos de los lados siempre coinciden con el radio de la circunferencia r, el área del polígono de n lados puede calcularse con la fórmula

$$a_n = \frac{1}{2} n r^2 F r sen\left(\frac{360}{n}\right)$$

a. Calcular el área de los polígonos de 10^i lados, para $i = 1, \ldots, 6$ tomando $r = 1$.

Solución

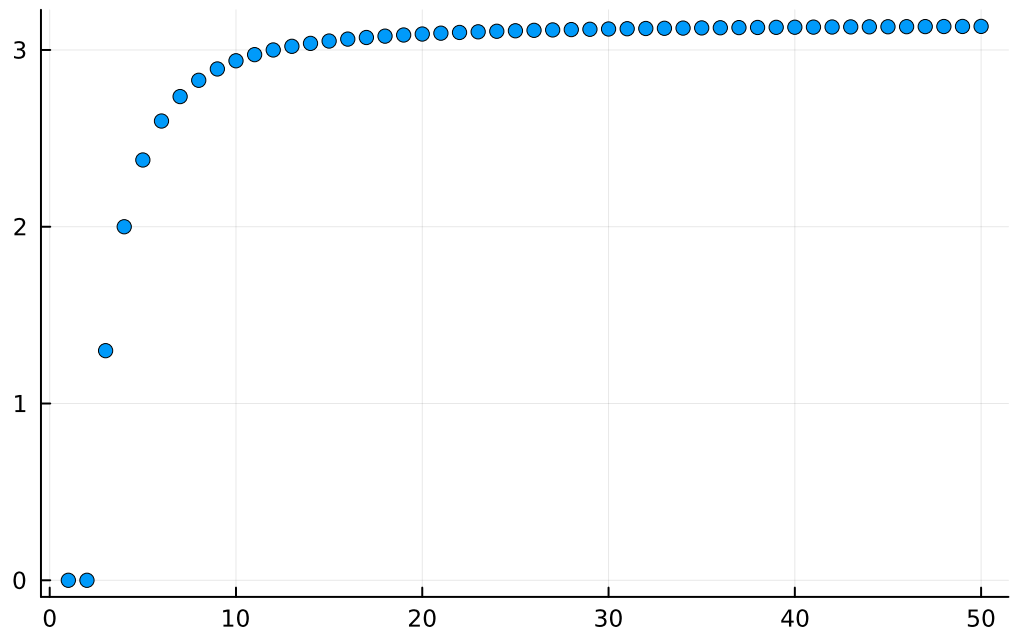
```
a(n) = n*sind(360/n)/2
print([a(10^i) for i = 1:6])
```

```
[2.938926261462366, 3.1395259764656687, 3.1415719827794755, 3.141592446881286, 3.141592
```

b. Dibujar con los primeros 50 términos de la sucesión de las areas de los polígonos tomando $r = 1$.

Solución

```
using Plots
a(n) = n*sind(360/n)/2
scatter([a(n) for n = 1:50], legend=false)
```



c. Calcular el límite de la sucesión de las areas de los polígonos tomando $r = 1$.

Solución

```
using SymPy
@syms n::(integer, positive)
a(n) = n*sin(2pi/n)/2
limit(a(n), n=>oo)
```

3.14159265358979

d. Usando el resultado anterior, calcular el area del círculo de radio r.

Solución

```
using SymPy
@syms n::(integer, positive), r
a(n) = n*r^2*sin(2pi/n)/2
limit(a(n), n=>oo)
```

$3.14159265358979r^2$

2.2 Ejercicios propuestos

Ejercicio 2.5. Calcular el décimo término de la sucesión $\left(\frac{3n^2+n}{6n^2-1}\right)_{n=1}^{\infty}$.

*Hint: *

Introducir hasta 5 decimales

Ejercicio 2.6. Calcular los 10 primeros términos de la sucesión $\left(\frac{3n^2+n}{6n^2-1}\right)_{n=1}^{\infty}$ y averiguar hacia dónde converge.

- ☐ 1
- ☐ 0.5
- ☐ No converge
- ☐ 0
- ☐ 1.5

Ejercicio 2.7. ¿Cuál de las siguientes gráficas corresponde a la sucesión $x_1 = 3$ y $x_{n+1} = \sqrt{2x_n}$ $\forall n = 2, 3, ...$

insert image here

Ejercicio 2.8. A la vista de la gráfica de los 20 primeros términos de la sucesión $\left(\frac{2^n}{n!}\right)_{n=1}^{\infty}$, ¿crees que la sucesión converge?

- ☐ Si
- ☐ No

Ejercicio 2.9. A la vista de la gráfica de los 10 primeros términos de la sucesión $\left(\frac{n^n}{n!}\right)_{n=1}^{\infty}$, ¿crees que la sucesión converge?

☐ Si

☐ No

Ejercicio 2.10. A la vista de la gráfica de los 20 primeros términos de la sucesión dada por $x_1 = 1$ y $x_{n+1} = \sqrt{x_n + 2}$ $\forall n \in \mathbb{N}$, ¿crees que la sucesión converge?

☐ Si

☐ No

Ejercicio 2.11. ¿Cuál es el límite de la sucesión $\left(\left(1 + \frac{2}{n}\right)^n\right)_{n=1}^{\infty}$

*Hint: *

Introducir hasta 5 decimales