

Procedimentos Graficos em Calculo Integral



Universidade Estadual de Maringá
Departamento de Matemática

Prof. Doherty Andrade (DMA- UEM)

Prof. Timothy M. (WLU-USA)

Maple

'E permitido copiar desde que citado a fonte. Contactos doherty@gauss.dma.uem.br

Sintaxe: `riemann2d(f,x=a..b);` ou,

`riemann2d(f,x=a..b,opts);`

PARAMETROS: `f` - uma função ou expressão,

`x` - a variavel de `f`,

`a,b` - numeros reais ($a < b$) especificando a variação,

`opts` - opcoes extra,

Resumo:

- O procedimento chama uma sequencia animada mostrando a soma de Riemann, dando a melhor aproximação e a área sob a curva.

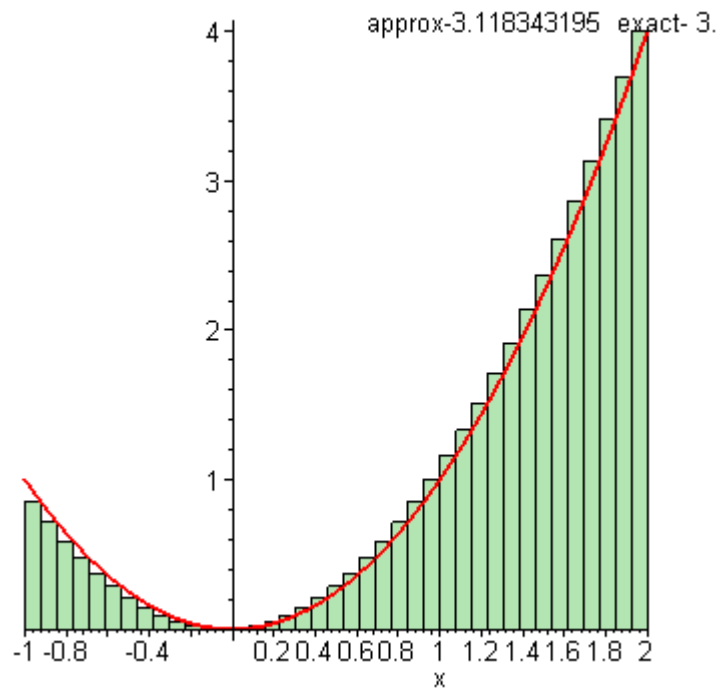
- Opções extra são `accuracy = m` onde `m` é algum real positivo e `frames = n` onde `n` é algum inteiro positivo especificando o número máximo de retângulos sob a função. Os defaults são `accuracy = .1`, `frames = 50`, e somas medias.

[Execute este procedimento e faça os exemplos.](#)

O Procedimento (execute-o)

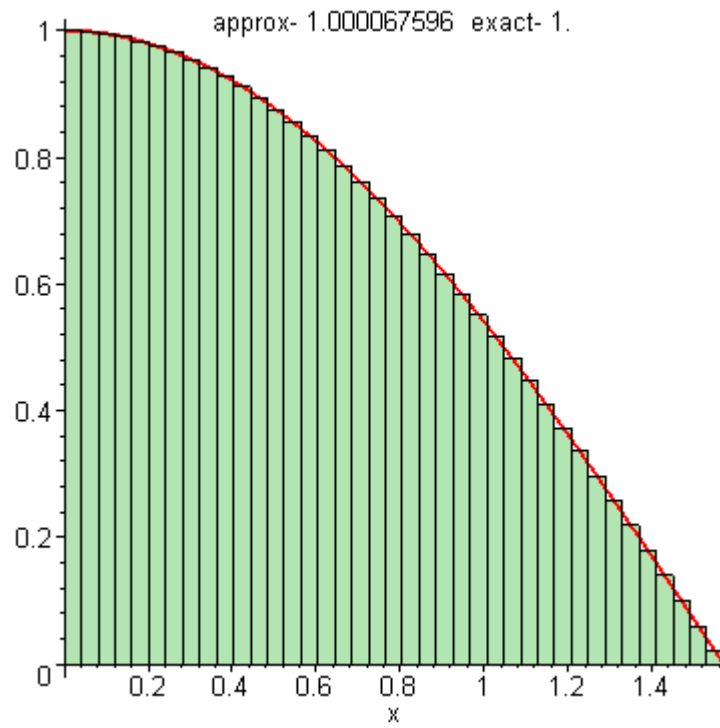
Exemplos

> `riemann2d(x^2,x=-1..2,frames=40,right);`



> `#riemann2d(x^3,x=0..2,frames=40,right);`

> `riemann2d(cos(x),x=0..Pi/2,frames=40,midle);`



>

O Procedimento (execute-o)

```
> riemann2d := proc ()
> local i,f, range, the_var, max_frames, acc, type_of_plot,
> v, the_value,L, n, num_frames, low_value, high_value, the_maximum;
> # standard assignments
> f := args[1];
> range := args[2];
> the_var := convert(range,list)[1];
> low_value := convert(convert(range,list)[2],list)[1];
> high_value := convert(convert(range,list)[2],list)[2];
> max_frames := 50;
>
> # defaults for options (will be this if no option given)
> acc:= .1;
> type_of_plot := middle;
> num_frames := 50;
>
> for i from 3 to nargs do
> if type(args[i],string) then
> if args[i]='middle' then type_of_plot:=middle fi;
> if args[i]='right' then type_of_plot:=right fi;
> if args[i]='left' then type_of_plot:=left fi;
> fi;
> if type(args[i],`= `) then
> if convert(args[i],list)[1]='frames' then
> if type(convert(args[i],list)[2],integer) then
```

```

> num_frames := convert(args[i],list)[2];
> acc := .00000001;
> else
> ERROR(`Number of frames must be an integer!`)
> fi;
> fi;
> if convert(args[i],list)[1]='accuracy' then
> acc := convert(args[i],list)[2];
> num_frames := 1000;
> fi;
> fi;
> od;
>
> # now for the hard part...
> # here is the actual value
> # assuming that the integral can be evaluated
>
> the_value := int(f,range);
>
> # decide where to put the values on the graph
> the_maximum := evalf(maximize(f,the_var,evalf(low_value)..evalf(high_value)));
>
> # here is the left plot
> if type_of_plot = left then
> v := student[leftsum](f,range,1);
> L := plots[display]({student[leftbox](f,range,1),plots[textplot]
> ((high_value-low_value)/2,the_maximum,cat(`approx-`,

```

```

> convert(evalf(v),string),` exact- ` ,convert(evalf(the_value),
> string))] ,align=ABOVE)):
>
> for n from 2
> while evalf(abs(evalf(value(v))-evalf(the_value)))>evalf(acc)
> do
> if evalf(n) > evalf(num_frames) or n = max_frames then break fi;
> v := student[leftsum](f,range,n);
> L := L,plots[display]({student[leftbox](f,range,n),plots[textplot]
> ([ (high_value-low_value)/2,the_maximum,cat(` approx- `,
> convert(evalf(v),string),` exact: ` ,convert(evalf(the_value),
> string))] ,align=ABOVE))):
> od;
> fi;
> # here is the middle plot
> if type_of_plot = middle then
> v := student[middlesum](f,range,1);
> L := plots[display]({student[middlebox](f,range,1),plots[textplot]
> ([ (high_value-low_value)/2,the_maximum,cat(` approx- `,
> convert(evalf(v),string),` exact- ` ,convert(evalf(the_value),
> string))] ,align=ABOVE))):
>
> for n from 2
> while evalf(abs(evalf(value(v))-evalf(the_value)))>evalf(acc)
> do
> if evalf(n) > evalf(num_frames) or n = max_frames then break fi;
> v := student[middlesum](f,range,n);

```

```

> L := L,plots[display]({student[middlebox](f,range,n),plots[textplot]
> ([[(high_value-low_value)/2,the_maximum,cat(`approx-`,
> convert(evalf(v),string),`exact-`,convert(evalf(the_value),
> string))] ,align=ABOVE)}):
> od;
> fi;
>
> # here is the right plot
> if type_of_plot = right then
> v := student[rightsum](f,range,1);
> L := plots[display]({student[rightbox](f,range,1),plots[textplot]
> ([[(high_value-low_value)/2,the_maximum,cat(`approx-`,
> convert(evalf(v),string),`exact-`,convert(evalf(the_value),
> string))] ,align=ABOVE)}):
>
> for n from 2
> while evalf(abs(evalf(value(v))-evalf(the_value)))>evalf(acc)
> do
> if evalf(n) > evalf(num_frames) or n = max_frames then break fi;
> v := student[rightsum](f,range,n);
> L := L,plots[display]({student[rightbox](f,range,n),plots[textplot]
> ([[(high_value-low_value)/2,the_maximum,cat(`approx-`,
> convert(evalf(v),string),`exact-`,convert(evalf(the_value),
> string))] ,align=ABOVE)}):
> od;
> fi;
> # finalmente...

```

```
> plots[display]([L],insequence=true);
```

```
>
```

```
> end:
```