

The Figure Class

Basic plotting in C++

Contents

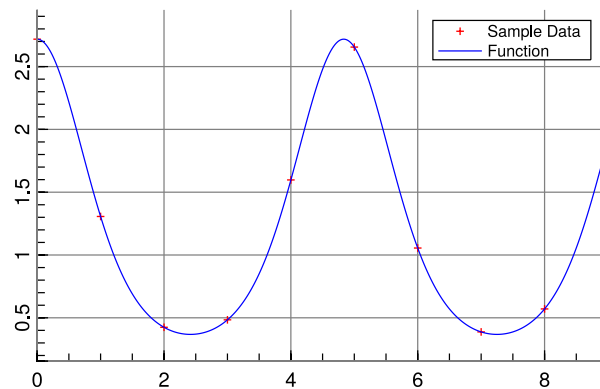
| | | |
|----------|-----------------------------|-----------|
| 1 | Opening example | 2 |
| 2 | Methods | 3 |
| 2.1 | grid | 3 |
| 2.2 | xlabel | 4 |
| 2.3 | ylabel | 4 |
| 2.4 | legend | 5 |
| 2.5 | setlog | 5 |
| 2.6 | plot | 6 |
| 2.7 | plot3 | 7 |
| 2.8 | fplot | 8 |
| 2.9 | ranges | 8 |
| 2.10 | save | 9 |
| 2.11 | title | 9 |
| 3 | Line characteristics | 10 |

1 Opening example

This short example code will show, how the Figure class can be used.

```
# include <vector>
# include <Eigen/Dense>
# include <figure/figure.hpp>

int main()
{
    std::vector<double> x(10), y(10);
    for (int i = 1; i <= 10; ++i){
        x[i] = i; y[i] = std::exp(std::cos(0.2*i));
    }
    Eigen::VectorXd u = Eigen::VectorXd::LinSpaced(500, 1, 10),
        v = ( (0.2*u.array()).cos().exp() ).matrix();
    mgl::Figure fig;
    fig.plot(x, y, " +r").label("Sample Data");
    fig.plot(u, v, " b").label("Function");
    fig.legend();
    fig.save("plot.eps");
    return 0;
}
```



2 Methods

2.1 grid

Definition:

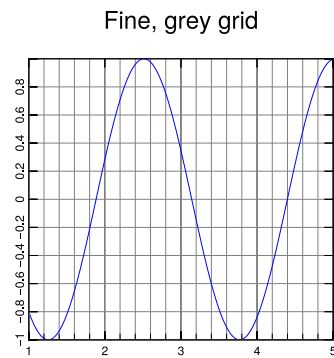
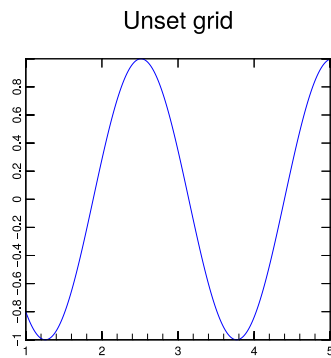
```
void grid( const bool& on = true,
           const std::string& gridType = "-",
           const std::string& gridCol = "h" )
```

Restrictions: None.

Examples:

```
mgl::Figure fig;
fig.plot(x, y);
fig.grid(false); // unset grid
fig.save("plot.eps");

mgl::Figure fig;
fig.plot(x, y);
fig.grid(true, "!", "h"); // grey fine mesh
fig.save("plot.eps");
```



2.2 xlabel

Definition:

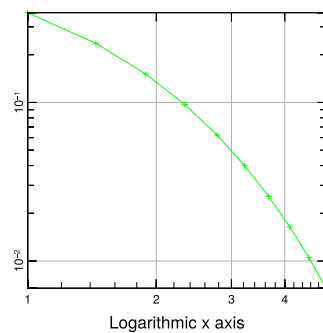
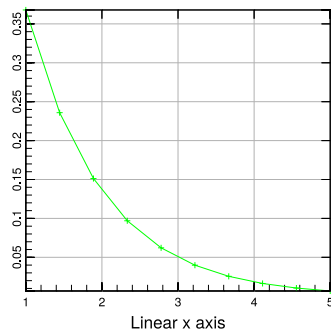
```
void xlabel( const std::string& label,
             const double& pos = 0 )
```

Restrictions: None.

Examples:

```
mg1::Figure fig;
fig.plot(x, y, "g+"); // 'g+' equals matlab/python '+-g'
fig.xlabel("Linear x axis");
fig.save("plot.eps");

mg1::Figure fig;
fig.xlabel("Logarithmic x axis"); // no restrictions on call order
fig.setlog(true, true);
fig.plot(x, y, "g+");
fig.save("plot.eps");
```



2.3 ylabel

Definition:

```
void ylabel( const std::string& label,
             const double& pos = 0 )
```

Restrictions: None.

Examples: See xlabel.

2.4 legend

Definition:

```
void legend( const double& xPos = 1,
             const double& yPos = 1 )
```

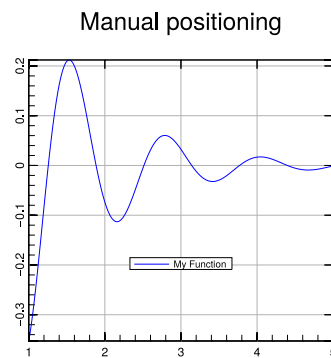
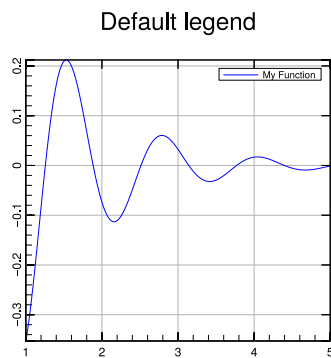
Restrictions: None.

Examples:

```
mgl::Figure fig;
fig.plot(x0, y0).label("My Function");
fig.legend(); // 'activate' legend
fig.save("plot");

mgl::Figure fig;
fig.plot(x0, y0).label("My Function");
fig.legend(0.5, 0.25); // set position to (0.5, 0.25)
fig.save("plot");

mgl::Figure fig;
fig.plot(x0, y0).label("My Function");
fig.save("plot"); // legend won't appear as legend() hasn't been called
```



2.5 setlog

Definition:

```
void setlog( const bool& logx = false,
             const bool& logy = false,
             const bool& logz = false )
```

Restrictions: All plots will use the latest `setlog` options or default if none have been set.

Examples:

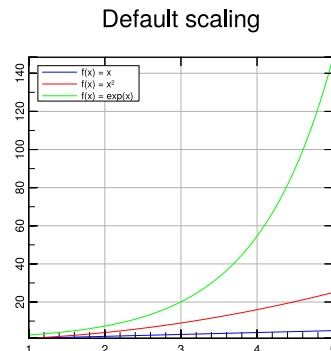
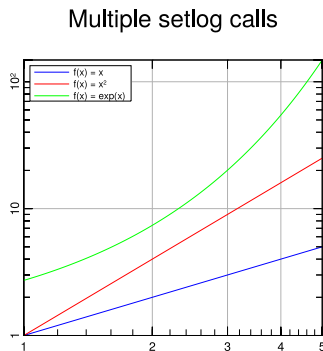
```
mgl::Figure fig;
fig.setlog(true, false); // -> semilogx
fig.plot(x0, y0);
fig.setlog(false, true); // -> semilogy
```

```

fig.plot(x1, y1);
fig.setlog(true, true); // -> loglog
fig.plot(x2, y2);
fig.save("plot.eps"); // ATTENTION: all plots will have been plotted in loglog-scale

mgl::Figure fig;
fig.plot(x, y);
fig.save("plot.eps"); // -> default (= linear) scaling

```



2.6 plot

Definition:

```

template <typename yVector>
void plot( const yVector& y,
           const std::string& style = "" )

template <typename xVector, typename yVector>
void plot( const xVector& x,
           const yVector& y,
           const std::string& style = "" )

```

Restrictions: `xVector` and `yVector` must have a `size()` method, which returns the size of the vector and a `data()` method, which returns a pointer to the first element in the vector. Furthermore `x` and `y` must have same length.

Examples:

```

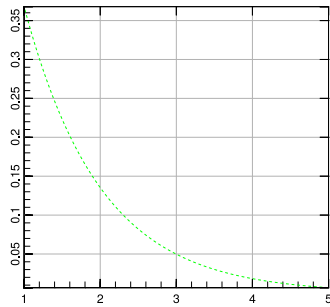
mgl::Figure fig;
fig.plot(x, y, "g;"); // green and dashed linestyle
fig.save("data.eps");

mgl::Figure fig;
fig.plot(x, y); // OK - style is optional
fig.save("data.eps");

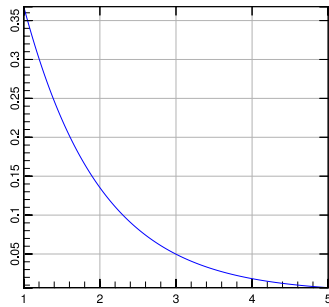
mgl::Figure fig;
fig.plot(x, y, " *r", "Data w/ red dots"); // ' *r' equals matlab/python 'r*'
fig.save("data.eps");

```

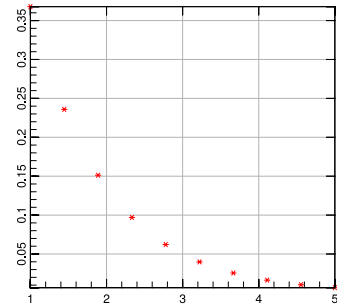
Plot w/ dashed style



Default style



Data plotting



2.7 plot3

Definition:

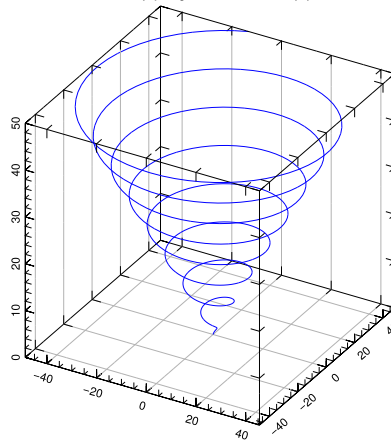
```
template <typename xVector, typename yVector, typename zVector>
void plot3( const xVector& x,
            const yVector& y,
            const zVector& z,
            const std::string& style = "" )
```

Restrictions: Same restrictions as in plot for two vectors, extended to zVector.

Examples:

```
mgl::Figure fig;
fig.plot3(x, y, z);
fig.save("trajectories.eps");
```

$$x = t \cdot \sin(t), y = t \cdot \cos(t), z = t$$



2.8 fplot

Definition:

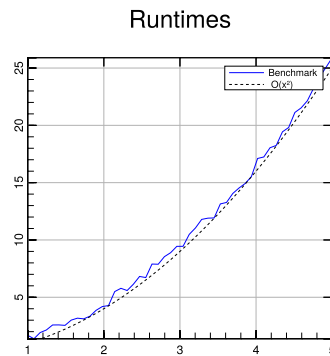
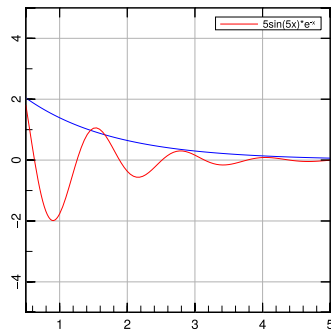
```
void fplot( const std::string& function,
            const std::string& style = "" )
```

Restrictions: None.

Examples:

```
mg1::Figure fig;
fig.fplot("(3*x^2 - 4.5/x)*exp(-x/1.3)");
fig.fplot("5*sin(5*x)*exp(-x)", "r").label("5sin(5x)*e^{-x}");
fig.ranges(0.5, 5, -5, 5); // be sure to set ranges for fplot!
fig.save("plot.eps");

mg1::Figure fig;
fig.plot(x, y, "b").label("Benchmark");
fig.fplot("x^2", "k;").label("\\ O(x^2)");
// here we don't set the ranges as it uses the range given by the x,y data
// and we use fplot to draw a reference line (O(x^2))
fig.save("runtimes.eps");
```



2.9 ranges

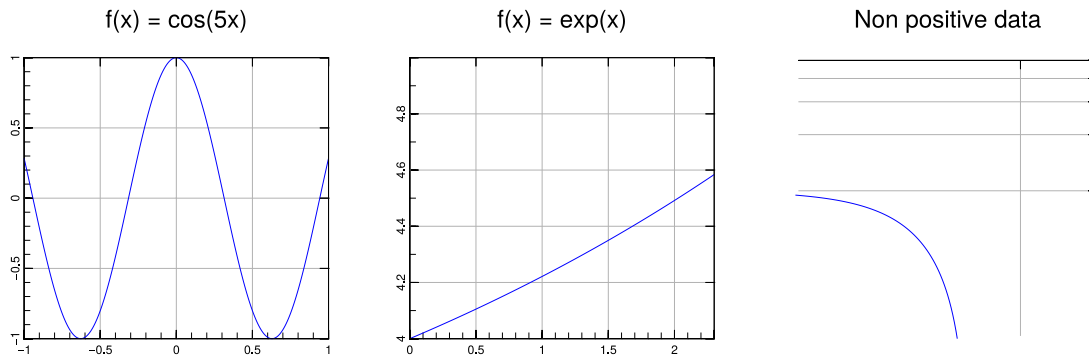
Definition:

```
void ranges( const double& xMin,
             const double& xMax,
             const double& yMin,
             const double& yMax )
```

Restrictions: $xMin < xMax$, $yMin < yMax$ and ranges must be > 0 for axis in logarithmic scale.

Examples:

```
mg1::Figure fig;
fig.ranges(-1,1,-1,1);
fig.plot(x, y, "b");
```

```

mgl::Figure fig;
fig.plot(x, y, "b");
fig.ranges(0, 2.3, 4, 5); // ranges can be called before or after 'plot'

mgl::Figure fig;
fig.ranges(-1, 1, 0, 5);
fig.setlog(true, true); // will run but MathGL will throw a warning
fig.plot(x, y, "b");

```

2.10 save

Definition:

```
void save( const std::string& file )
```

Restrictions: Supported file formats: .eps and .png.

Examples:

```

mgl::Figure fig;
fig.save("plot.eps"); // OK

mgl::Figure fig;
fig.save("plot"); // OK - will be saved as plot.eps

mgl::Figure fig;
fig.save("plot.png"); // OK - but needs -lpng flag!

```

2.11 title

Definition:

```
void title( const std::string& text )
```

Restrictions: None.

3 Line characteristics

Linecolors^a:

| | |
|--------------|---|
| blue | b |
| green | g |
| red | r |
| cyan | c |
| magenta | m |
| yellow | y |
| gray | h |
| green-blue | l |
| sky-blue | n |
| orange | q |
| green-yellow | e |
| blue-violet | u |
| purple | p |

^a Upper-case letters will give a darker version of the lower-case version.

Linestyles:

| | |
|-------------------|---|
| none | |
| solid | - |
| dashed | ; |
| small dashed | = |
| long dashed | |
| dotted | : |
| dash-dotted | j |
| small dash-dotted | i |

None is used as follows:
" r*" gives red stars w/o
any lines

Linemarkers:

| | |
|---|----|
| + | + |
| o | o |
| ◇ | d |
| · | · |
| △ | ^ |
| ▽ | v |
| < | < |
| > | > |
| ⊙ | #. |
| ⊕ | #+ |
| ⊗ | #x |