Cubic B-splines Using PSTricks

Michael Sharpe

msharpe@ucsd.edu

A cubic uniform B-spline curve with control points $B_0 \dots B_n$ is a curve parametrized by the interval $[0, n]$, which is C^2-continuous (that is, has continuous curvature) and is on each interval $[k-1, k]$ given by a cubic Bézier curve whose control points are derived from the (B_k). These curves are discussed in any reasonably modern text on Numerical Analysis. One easily accessible source is the UCLA lecture notes of Kirby Baker:

`http://www.math.ucla.edu/~baker/149.1.02w/handouts/dd_splines.pdf`

I'll focus on two special cases: (i) relaxed, uniform B-splines; (ii) periodic, uniform B-splines. Uniform refers to the condition mentioned in the first paragraph: each Bézier sub-curve is parametrized by an interval of length 1. Relaxed means that the curvature at the endpoints $t = 0, t = n$ is zero. Periodic means in effect that the B_i repeat periodically, and the curve generated is a closed curve.

1 Relaxed, Open B-spline

The algorithm has the following steps.

- The curve starts at B_0 and ends at B_n.
- Divide each line $B_{k-1}B_k$ into equal thirds, with subdivision points labeled R_{k-1}, L_k respectively, so that B_k has L_k as its immediate neighbor to the left, and R_k as its immediate neighbor to the right.
- For $0 < k < n$, divide the line segment L_kR_k in half, letting S_k denote the midpoint. In effect, for $0 < k < n$, $S_k = (B_{k-1} + 4B_k + B_{k+1})/6$.
- Let $S_0 = B_0$ and $S_n = B_n$.
- For $0 < k \leq n$, construct the cubic Bézier curve with control points S_{k-1}, R_{k-1}, L_k, S_k, parametrized by $k - 1 \leq t \leq k$.

The `pst-Bspline` package implements this algorithm as `\psBspline`, whose simplest form is, for example

```
\psBspline(.5,.5)(2,0)(5,2)(6,4)(4,5)(2,4)
```

The coordinates are the B-spline control points. Aside from the usual keywords, like `linestyle`, `linecolor` and `arrows`, there is a Boolean keyword `showframe`. The effect of `showframe=true` is to show the intermediate points and lines in the algorithm described above.

There is another optional argument that can be applied if you wish to be able to refer to any of the points constructed in the algorithm. By example,

```
\psBspline{B}(.5,.5)(2,0)(5,2)(6,4)(4,5)(2,4)
```

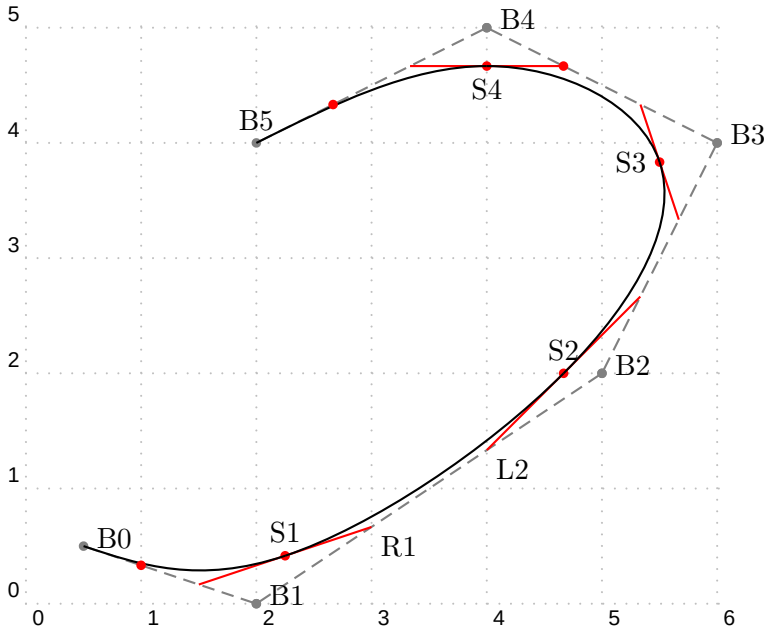
sets the root of the naming scheme to `B`, the effect of which is that the B-spline control points will be nodes of type `\pnode` with names `B0`, `B1` and so on, the other points being similarly named `BL0`, `BL1`, ... , `BR0`, `BR1`, ... , `BS0`, `BS1`, ... For example, to draw a line between `BL1` and `BS4`, just use `\ncline(BL1)(BS4)`.

The algorithm is implemented entirely in PSTricks code with PostScript specials, but no PostScript header file, depending for the most part on the flexibility of nodes, and above all the `\multido` macro, which allows one to construct with relative ease items that look and feel like arrays. Use of `\SpecialCoor` is essential.

There is a closely related macro `\psBsplineE` which removes the first and last Bézier segments, much as `\psecurve` acts in relation to `\pscurve`, allowing one one to draw B-splines with non-zero curvature at the endpoints.

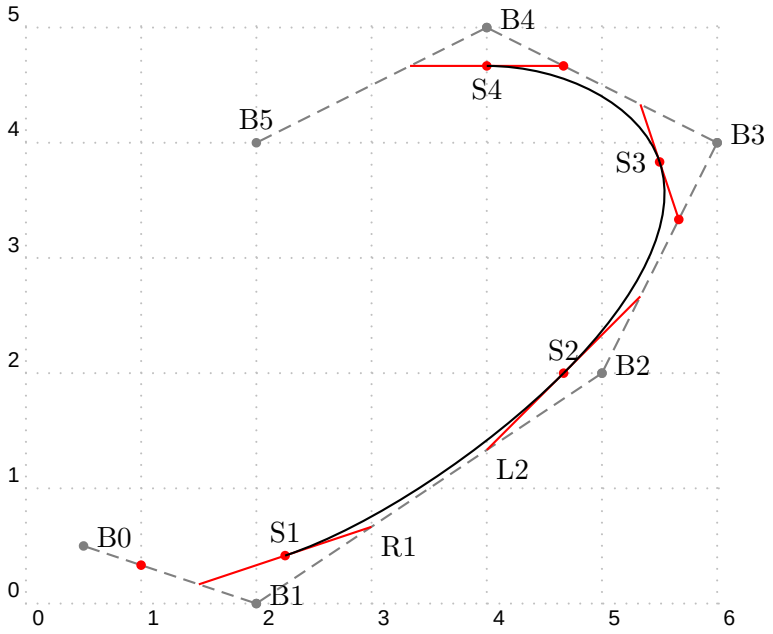
```
\documentclass{article}
\usepackage{pstricks}
\usepackage{multido,pst-node,pst-bspline}
\pagestyle{empty}
\begin{document}
\SpecialCoor % essential for pst-bspline package
\psset{unit=.6in}
\begin{pspicture}[showgrid=true](-.5,-.5)(6,5)
\psBspline[showframe=true]{B}(.5,.5)(2,0)(5,2)(6,4)(4,5)(2,4)
\multido{\i=0+1}{5}{\uput[20](B\i){B\i}}
\uput[90](B5){B5}
\uput[90](BS1){S1}
\uput[90](BS2){S2}
\uput[180](BS3){S3}
\uput[270](BS4){S4}
\uput[-45](BR1){R1}
```

```
\uput[-45](BL2){L2}
\end{pspicture}
\end{document}
```




```
\documentclass{article}
\usepackage{pstricks}
\usepackage{multido,pst-node,pst-bspline}
\pagestyle{empty}
\begin{document}
\SpecialCoor % essential for pst-bspline package
\psset{unit=.6in}
\begin{pspicture}[showgrid=true](-.5,-.5)(6,5)
\psBsplineE[showframe=true]{B}(.5,.5)(2,0)(5,2)(6,4)(4,5)(2,4)
\multido{\i=0+1}{5}{\uput[20](B\i){B\i}}
\uput[90](B5){B5}
\uput[90](BS1){S1}
\uput[90](BS2){S2}
\uput[180](BS3){S3}
\uput[270](BS4){S4}
\uput[-45](BR1){R1}
\uput[-45](BL2){L2}
```

```
\end{pspicture}
\end{document}
```



2 Periodic B-spline

The result here is a closed curve. The algorithm is essentially the same as in the preceding case, except:

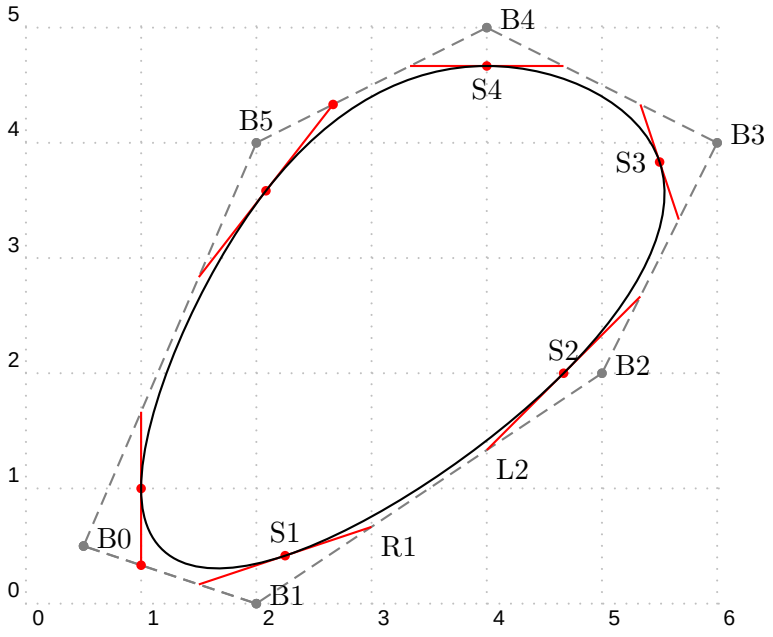
- Extend B_i periodically with period $n+1$, so that $B_{n+1} = B_0$ and $B_{n+2} = B_1$.
- Construct R_i, L_i for $0 < i < n+2$, as above.
- Construct S_k as above (midpoint of $L_k R_k$), for $0 < k < n+2$.
- Set $S_0 = S_{n+1}$.
- For $0 < k \leq n+1$, construct the cubic Bézier curve with control points S_{k-1}, R_{k-1}, L_k, S_k, parametrized by $k-1 \leq t \leq k$.

The macro in this case is `\psBsplineC`, where the `C` stands for Closed. The code, being implemented as a `\pscustom` object, does not accept the

`doubleline` keyword, but does accept, for example,

```
fillstyle=solid,fillcolor=gray
```

```
\documentclass{article}
\usepackage{pstricks}
\usepackage{multido,pst-node,pst-bspline}
\pagestyle{empty}
\begin{document}
\SpecialCoor % essential for pst-bspline package
\psset{unit=.6in}
\begin{pspicture}[showgrid=true](-.5,-.5)(6,5)
\psBsplineC[showframe=true]{B}(.5,.5)(2,0)(5,2)(6,4)(4,5)(2,4)
\multido{\i=0+1}{5}{\uput[20](B\i){B\i}}
\uput[90](B5){B5}\uput[90](BS1){S1}
\uput[90](BS2){S2}\uput[180](BS3){S3}
\uput[270](BS4){S4}\uput[-45](BR1){R1}
\uput[-45](BL2){L2}
\end{pspicture}
\end{document}
```



3 Related constructions

There are in addition three additional macros that draw similar curves, but organized in a slightly different way. They are particularly useful when there is a sequence of points already defined as `\pnode`s. Here is a simple way to define such a sequence.

3.1 The `pnodes` macro

The line

```
\pnodes{P}(2,1.5)(3,4)(5,1)
```

defines a sequence of `\pnode`s with the node root `P`: `P0`=(2,1.5), `P1`=(3,4) and `P0`=(5,1). The sequence may be any (reasonable) length. The macro leaves an entry in the console saying that it has defined nodes `P0 .. P2`. The three new macros are:

```
\psBsplineNodes{<node root>}{<top index>}
\psBsplineNodesC{<node root>}{<top index>}
\psBsplineNodesE{<node root>}{<top index>}
```

corresponding to the macros `\psBspline`, `\psBsplineC` and `\psBsplineE`. The difference is that the macros with `Nodes` in the name have as arguments the root node name and the last index, rather than the list of points. For example, with the above definition of `P` in force, `\psBsplineNodes{P}{2}` has exactly the same effect as `\psBspline(2,1.5)(3,4)(5,1)`.

4 B-spline Interpolation

This is the inverse problem. Being given points $(S_k)_{0\leq k\leq n}$, the goal is to produce the B-spline control points B_k leading to the points S_k, so that the associated B-spline curve interpolates the S_k.

4.1 Open curve

We discuss first the case of an open, uniform B-spline curve with relaxed endpoints. According to the discussion above, we have to solve the equa-

tions

$$\begin{aligned} B_0 &= S_0 \\ B_0 + 4B_1 + B_2 &= 6S_1 \\ B_1 + 4B_2 + B_3 &= 6S_2 \\ &\cdots \\ B_{n-2} + 4B_{n-1} + B_n &= 6S_{n-1} \\ B_n &= S_n \end{aligned}$$

for the B_k. In matrix form, this becomes the tridiagonal system

$$\begin{pmatrix} 4 & 1 & & & \\ 1 & 4 & 1 & & \\ & 1 & 4 & 1 & \\ & & \cdots & & 1 \\ & & & 1 & 4 \end{pmatrix} \begin{pmatrix} B_1 \\ B_2 \\ B_3 \\ \cdots \\ B_{n-1} \end{pmatrix} = \begin{pmatrix} 6S_1 - S_0 \\ 6S_2 \\ 6S_3 \\ \cdots \\ 6S_{n-1} - S_n \end{pmatrix}$$

The LU decomposition of the tridiagonal matrix may be seen to take the form

$$\begin{pmatrix} 1 & & & & \\ m_1 & 1 & & & \\ & m_2 & 1 & & \\ & & \cdots & & \\ & & & m_{n-2} & 1 \end{pmatrix} \begin{pmatrix} m_1^{-1} & 1 & & & \\ & m_2^{-1} & 1 & & \\ & & m_3^{-1} & 1 & \\ & & & \cdots & 1 \\ & & & & m_{n-1}^{-1} \end{pmatrix}$$

where $m_1 = 1/4$, $m_{k+1} = 1/(4 - m_k)$ for $k = 1, \cdots, n-2$. The solution of the original system is therefore accomplished in two steps, introducing intermediate points (R_k), by (in pseudo-code)

```
R_1=6*S_1-S_0
for i=2 to n-2
  R_i=6*S_i-m_{i-1}* R_{i-1}
R_{n-1}=(6*S_{n-1}-S_n)-m_{n-2}*R_{n-2}
B_{n-1}=m_{n-1}*R_{n-1}
for i=n-2 downto 1
  B_i=m_i*(R_i-B_{i+1})
```

The code for the `\psBsplineInterp` command uses this algorithm to solve for the B_k as nodes, except that in order to save node memory, the B nodes are substituted in place for the R nodes, so that, for example, the first step becomes `B_1=6*S_1-S_0`.

Assuming you have previously defined nodes `S0` $\cdots$ `S4`,

```
\psBsplineInterp{S}{4}
```

will construct a sequence `SB0` $\cdots$ `SB4` of nodes at the B-spline control points for the relaxed, uniform cubic B-spline interpolating the `Sk`, and this curve may then be rendered with the command

```
\psBsplineNodes{SB}{4}
```

If you don't care about keeping track of the internal operations and names for nodes, you may generate the curve directly with, for example,

```
\psbspline(0,0)(.5,.1)(1.5,.6)(2.5,1.4)(3.5,1.8)(4.5,1.7)%
(5.8,1.0)(7.5,.25)(10,0)
```

4.2 Closed (periodic) case

We turn now to the periodic uniform B-spline curve interpolating n points S_0,...,S_{n-1}. Extend the sequence periodically with period n, so that $S_n = S_0$, $S_{n+1} = S_1$, $S_{-1} = S_{n-1}$, and so on. In order to find the periodic control points B_k, we have to solve the n equations

$$\begin{gathered} B_n + 4B_1 + B_2 = 6S_1 \\ B_1 + 4B_2 + B_3 = 6S_2 \\ \cdots \\ B_{n-2} + 4B_{n-1} + B_n = 6S_{n-1} \\ B_{n-1} + 4B_n + B_1 = 6S_n \end{gathered}$$

for the B_k, $1 \le k \le n$. In matrix form, this becomes the system

$$\begin{pmatrix} 4 & 1 & & & 1 \\ 1 & 4 & 1 & & \\ & 1 & 4 & 1 & \\ & & \cdots & & 1 \\ 1 & & & 1 & 4 \end{pmatrix} \begin{pmatrix} B_1 \\ B_2 \\ B_3 \\ \cdots \\ B_n \end{pmatrix} = \begin{pmatrix} 6S_1 \\ 6S_2 \\ 6S_3 \\ \cdots \\ 6S_n \end{pmatrix}$$

Let $(x_k, y_k) = 6S_k$. We perform Gaussian elimination on the matrix

$$\begin{pmatrix} 4 & 1 & & & 1 & x_1 & y_1 \\ 1 & 4 & 1 & & & x_2 & y_2 \\ & 1 & 4 & 1 & & x_3 & y_3 \\ & & \cdots & & 1 & & \\ 1 & & & 1 & 4 & x_n & y_n \end{pmatrix}$$

As in the previous case, let $m_1 = 0.25$, $m_k = 1/(4 - m_{k-1})$ for $k \geq 2$. The factor m_k will be the multiplier of row k after the previous row operation, in order to normalize the row. These are the steps in the procedure.

- Initialize: multiply row 1 by m_1 so that its first entry (1,1) is 1. Replace x_1 by $m_1 x_1$ and y_1 by $m_1 y_1$. Entry $(1, n)$ is m_1.
- Subtract new row 1 from row 2 and multiply the resulting row by m_2. The leading entry (2,1) becomes 1. Entry $(2, n)$ becomes $-m_1 m_2$, and x_2, y_2 are updated to $m_2(x_2 - x_1)$, $m_2(y_2 - y_1)$. The superdiagonal entry (2,3) is the only other non-zero entry, and its new value is m_2.
- Subtract new row 1 from row n, so that its leading entry $(n, 2)$ is $-m_1$.
- Subtract new row 2 from row 3 and multiply the result by m_3. The leading entry (3,3) becomes 1 and the entry $(3, n)$ becomes $m_1 m_2$, with x_3, y_3 updating to $m_3(x_3 - x_2)$, $m_3(y_3 - y_2)$. The superdiagonal entry (3,4) is now m_3.
- Subtract new row 2 times $-m_1$ from row n, whose leading entry $(n, 3)$ is now $m_1 m_2$.
- Continue in this way until row $n-2$ has been subtracted as above from row $n-1$, multiplying the result by m_{n-1}, and a suitable multiple has been subtracted from row n. The leading entry of row $n-1$ (column $n-1$) is 1 and its n^{th} entry is $1-(-1)^n m_1 \cdots m_{n-2}$. Row n has leading entry in column $n-1$, equal to 1.
- Finally, subtract an appropriate multiple of row $n-1$ from row n so that row n has leading entry in column n. The resulting matrix is upper triangular, and we may now substitute back starting from the last row to give a complete reduction.

Here are the steps in pseudocode. We keep track of row n with the array b_k, column n with the array c_k. The indices for both run from 1 to n.

```
m(1)=0.25
for k=2 to n-1
   m(k)=1/(4-m(k-1))
b(1)=1
b(n-1)=1
b(n)=4
c(n-1)=1% don't need c(n), =b(n)
%multiply first row by m1
```

```
c(1)=m(1)
x(1)=m(1)*x(1)
y(1)=m(1)*y(1)
for k=2 to n-1
  %subtract normalized row k-1 from row k, renormalize row k
  c(k)=m(k)*(c(k)-c(k-1))%note that initially, c(k)=0 for 1<k<n-1
  x(k)=m(k)*(x(k)-x(k-1))
  y(k)=m(k)*(y(k)-y(k-1))
  %subtract normalized row k-1 times b(k-1) from row n
  b(k)=b(k)-b(k-1)*m(k-1)
  b(n)=b(n)-c(k-1)*b(k-1)
  x(n)=x(n)-x(k-1)*b(k-1)
  y(n)=y(n)-y(k-1)*b(k-1)
% subtract row n-1 times b(n-1) from row n, renormalize by 1/b(n)
b(n)=b(n)-b(n-1)*c(n-1)
x(n)=(x(n)-x(n-1)*b(n-1))/b(n)
y(n)=(y(n)-y(n-1)*b(n-1))/b(n)
%work back
x(n-1)=x(n-1)-c(n-1)*x(n)
y(n-1)=y(n-1)-c(n-1)*y(n)
for k=n-2 downto 1
   x(k)=x(k)-m(k)* x(k+1)-c(k)*x(n)
   y(k)=y(k)-m(k)* y(k+1)-c(k)*y(n)
```

This algorithm is implemented in TEX/PostScript code in `pst-Bspline.tex` and may be invoked using the macro

`\psBsplineInterpC{<node root>}{<index>}`

You must previously have defined a sequence, say `S0` $\cdots$ `S100` of `\pnode`s that you plan to interpolate with a closed curve. Then

`\psBsplineInterpC{S}{100}`

constructs the sequence `SB0` $\cdots$ `SB100` of B-spline control points (appending `B` to the root name) for a closed curve interpolating `S0` $\cdots$ `S100`, which may then be rendered with the command

`\psBsplineNodesC{SB}{100}`

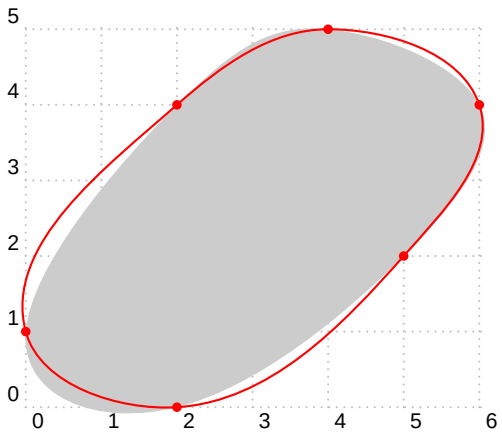
with any keywords options you wish.

The following example illustrates that there is a difference between `\psccurve` and B-spline interpolation, the former having a rounder appearance. Gener-

ally speaking, B-spline interpolation comes closer to minimizing the average curvature.

```
\documentclass{article}
\usepackage{pstricks}
\usepackage{pst-bspline,pstricks-add}
\begin{document}
\begin{pspicture}[showgrid=true](-.5,-.5)(6,5)
\pnodes{P}(0,1)(2,0)(5,2)(6,4)(4,5)(2,4)%
\psBsplineInterpC{P}{5}%
\psBsplineNodesC*[linecolor=gray!40]{PB}{5}%
\psccurve[linecolor=red,showpoints=true](0,1)(2,0)(5,2)(6,4)(4,5)(2,4)
\end{pspicture}
\end{document}
```

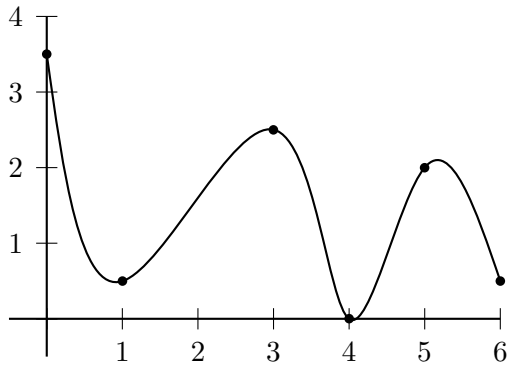
Slight difference between psccurve and B-spline interpolation



A B-spline curve can in many cases provide a good function interpolation mechanism, but the result is not guaranteed to be the graph of a function.

```
\begin{center}
\begin{pspicture}[showgrid=true](-.5,-.5)(6,4)
\psdots(0,3.5)(1,.5)(3,2.5)(4,0)(5,2)(6,.5)
\psaxes(0,0)(-.5,-.5)(6,4)
\psbspline(0,3.5)(1,.5)(3,2.5)(4,0)(5,2)(6,.5)
\end{pspicture}
```

```
\end{center}
```




```
\documentclass{article}
\usepackage{graphicx}
\usepackage{pstricks}
\usepackage{pst-bspline,pstricks-add}
\begin{document}
\psset{unit=.25in}
\begin{pspicture}[showgrid=true](-.5,-.5)(6,5)
\pnodes{P}(0,1)(2,0)(5,2)(6,4)(4,5)(2,4)
\pnode(3,3){C}
\multido{\ra=0+.05,\rb=1+.05,\i=30+1}{40}{%
\psBsplineC*[linecolor=blue!\i!brown]{B}%
([nodesep=\ra]{C}P0)([nodesep=\ra]{C}P1)%
([nodesep=\ra]{C}P2)([nodesep=\ra]{C}P3)%
([nodesep=\ra]{C}P4)([nodesep=\ra]{C}P5)}
\end{pspicture}
\end{document}
```