データ処理：表計算ソフトと同じことをしてみよう

*Mathematica*で表計算ソフトに出来ることを任せる方法を学ぼう.

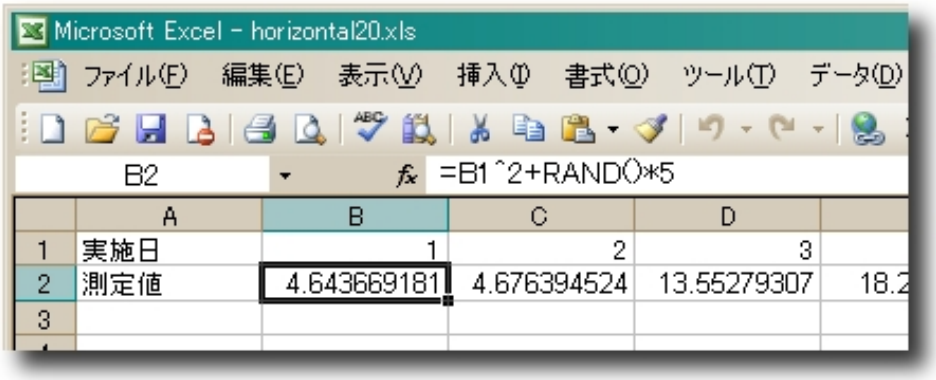
数や文字列を*Mathematica*に取り込んでみよう

- **ExcelやCalcなどの表計算ソフトからデータを取り込む**

表計算ソフトはとても便利なので，様々なデータを表計算ソフトのファイル形式で保存している人も多いのではないかと思います．ここでは，そのような表計算ソフトのファイルから数や文字列を*Mathematica*に読み込む（*Mathematica*が扱えるようにする）方法について説明したいと思います.

わかりやすく，次のような実施日と測定値のデータを表計算ソフトで作成したとしましょう．*Mathematica*はMicrosoft OfficeのExcelで保存したファイル（拡張子は「.xls」），StarSuiteやOpenOfficeのCalcで保存したファイル（拡張子は「.ods」ないしは「.sxc」）を読み込むことが出来ます.

簡単のため，実施日は1から20まで，測定値は図にあるような数式で生成したものとします．下記の説明ではファイル名を「horizontal20.xls」としていますが，必要に応じて実際のファイル名で読みかえるようにしてください.



表計算ソフト：サンプルExcelファイルの内容

*Mathematica*に「このファイルのデータを読み込みなさい」と指示を出すには「Import」という命令を使います．命令の出し方は非常に簡単で，次のように読み込みたいファイル名を文字列として指示するだけです．読み込んだデータには名前を付けておきましょう．ファイル名を入力するのが面倒な場合は，メニューの「挿入/ファイルパス...」を使うことで，普段慣れ親しんでいるファイル選択ダイアログから選択することも可能です（お勧めです）.

```
In[2]:= 横形式の生データ = Import["horizontal20.xls"]

Out[2]= {{{実施日, 1., 2., 3., 4., 5., 6., 7., 8., 9., 10.,
     11., 12., 13., 14., 15., 16., 17., 18., 19., 20.},
    {測定値, 5.02992, 5.11066, 13.0721, 19.4117, 27.3183,
     39.5797, 53.2928, 68.7363, 85.1994, 103.002,
     121.295, 148.34, 172.29, 197.446, 229.292,
     256.515, 291.443, 324.397, 363.773, 404.649}}}
```

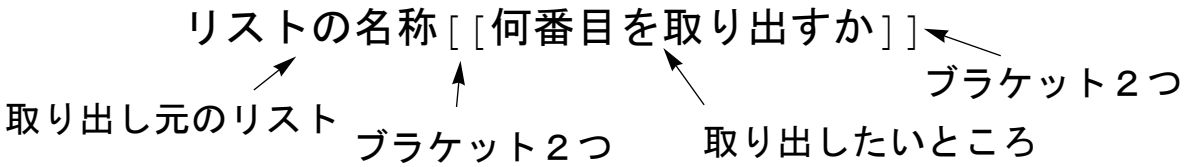
*Mathematica*で表計算ソフトのファイルがどのように認識されているかを確認してみましょう．「リストの構造を調べよ」という意味の「Dimensions」という命令を使う

```
In[3]:= Dimensions[横形式の生データ]

Out[3]= {1, 2, 21}
```

「{1, 2, 21}」という結果になりました．この数字の意味は「要素を21個持つリスト，を要素として2個持つリスト，を要素として1個持つリスト」です．表計算ソフトの用語では「シートの枚数は1つで，2行21列のデータ」となります．

このままでは扱いづらいので，リストの操作について少し勉強しましょう．*Mathematica*では，リストの直後にブラケット2つの組（「[[」と「]]」）で数字



リスト操作：データの一部分だけを取り出す

例えば，横形式の生データの一番外側のリストは「シート」を表していますから，次のようにブラケット2つの組で1番目を取り出すと，1シート目のデータだけを抜き出すことが出来ます．結果のリストの構造が変わっているのに気がついたでしょう

```
In[4]:= 最初のシート = 横形式の生データ[[1]]

Out[4]= {{実施日, 1., 2., 3., 4., 5., 6., 7., 8., 9., 10.,
     11., 12., 13., 14., 15., 16., 17., 18., 19., 20.},
    {測定値, 5.02992, 5.11066, 13.0721, 19.4117, 27.3183,
     39.5797, 53.2928, 68.7363, 85.1994, 103.002,
     121.295, 148.34, 172.29, 197.446, 229.292,
     256.515, 291.443, 324.397, 363.773, 404.649}}
```

```
In[5]:= Dimensions[最初のシート]
Out[5]= {2, 21}
```

更に，最初のシートの1番目（即ち，1行目）を取り出すことで，実施日のデータだけを抜き出すことが出来ます．この操作は「一番最初の要素を抜き出せ」という意味の

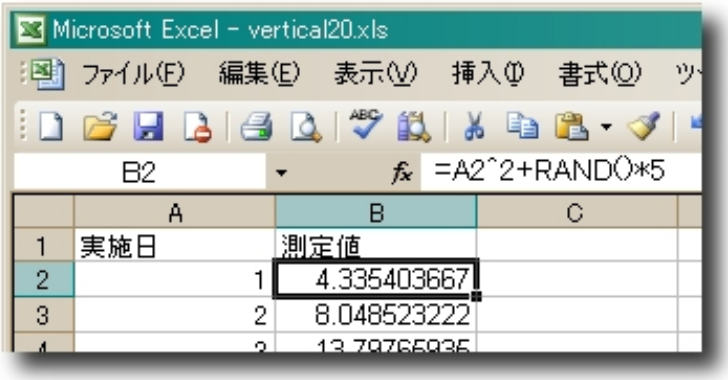
```
In[6]:= 実施日データ = 最初のシート[[1]]
Out[6]= {実施日, 1., 2., 3., 4., 5., 6., 7., 8., 9., 10.,
         11., 12., 13., 14., 15., 16., 17., 18., 19., 20.}
```

```
In[7]:= First[最初のシート]
Out[7]= {実施日, 1., 2., 3., 4., 5., 6., 7., 8., 9., 10.,
         11., 12., 13., 14., 15., 16., 17., 18., 19., 20.}
```

○○番目の中の○○番目という指定も出来ます．例えば，次のようにカンマで区切ることで「1番目の中の2番目」を取り出せます．このデータの場合，最初のシートの2行

```
In[8]:= 測定値データ = 横形式の生データ[[1, 2]]
Out[8]= {測定値, 5.02992, 5.11066, 13.0721, 19.4117,
         27.3183, 39.5797, 53.2928, 68.7363, 85.1994,
         103.002, 121.295, 148.34, 172.29, 197.446, 229.292,
         256.515, 291.443, 324.397, 363.773, 404.649}
```

表計算ソフトで作ったファイルの内容が下図にあるように縦に並んでいる場合も同じように読み込むことが出来ます．下記の説明ではファイル名を「vertical20.xls」として



表計算ソフト：サンプルExcelファイルの内容

横形式のデータと同じくImportという命令に，次のように読み込みたいファイル名を文字列として指示して読み込みます．読み込んだデータには名前を付けておきましょ

```
In[9]:= 縦形式の生データ = Import["vertical20.xls"]

Out[9]= {{{実施日, 測定値}, {1., 4.77725}, {2., 5.55288},
    {3., 9.32646}, {4., 19.5483}, {5., 27.6226},
    {6., 40.3695}, {7., 50.6452}, {8., 67.0622},
    {9., 81.9948}, {10., 102.578}, {11., 121.64},
    {12., 144.626}, {13., 170.994}, {14., 198.418},
    {15., 225.766}, {16., 259.499}, {17., 292.716},
    {18., 328.8}, {19., 365.317}, {20., 402.559}}}
```

表計算ソフトでデータを横に伸ばしているか，それとも縦に伸ばしているかによって，*Mathematica*内での表現も変化します．そこで，今回もDimensionsで構造を確認

```
In[10]:= Dimensions[縦形式の生データ]

Out[10]= {1, 21, 2}
```

「{1, 21, 2}」という数字の順番が異なる結果になりました．この数字の意味は「要素を2個持つリスト，を要素として21個持つリスト，を要素として1個持つリスト」です．表計算ソフトの用語では「シートの枚数は1つで，21行2列のデータ」となります．

多少構造は異なりますが，先ほどと同じようにブラケット2つの組で最初のシートだけを取り出せます．この部分取り出しは，どのようなリストに対しても行うことが出来

```
In[11]:= 最初のシート = 縦形式の生データ[[1]]

Out[11]= {{実施日, 測定値}, {1., 4.77725}, {2., 5.55288},
    {3., 9.32646}, {4., 19.5483}, {5., 27.6226},
    {6., 40.3695}, {7., 50.6452}, {8., 67.0622},
    {9., 81.9948}, {10., 102.578}, {11., 121.64},
    {12., 144.626}, {13., 170.994}, {14., 198.418},
    {15., 225.766}, {16., 259.499}, {17., 292.716},
    {18., 328.8}, {19., 365.317}, {20., 402.559}}
```

縦方向に延びているデータの場合，最初のシートの1番目を取り出すと，次のように項目名のペアになっています．なぜ，このペアが取り出されたかは，直前に問い合わせた最初のシートの1番目が項目名のペアのリストだからですね．2番目を取り出せ

```
In[12]:= 最初のシート[[1]]

Out[12]= {実施日, 測定値}
```

In[13]:= **最初のシート[[2]]**

Out[13]= {1., 4.77725}

このような縦方向に延びるデータの場合に，実施日だけ，または測定値だけのリストを取り出すにはどうしたら良いでしょうか．*Mathematica*には，このような場合に備えて，「全部」という意味の「All」という指示が出せるようになっています．例えば次のような指示は「全ての要素，ただし各要素の1番目だけを取り出せ」という意味に

In[14]:= **最初のシート[[All, 1]]**

Out[14]= {実施日, 1., 2., 3., 4., 5., 6., 7., 8., 9., 10.,
11., 12., 13., 14., 15., 16., 17., 18., 19., 20.}

In[15]:= **最初のシート[[All, 2]]**

Out[15]= {測定値, 4.77725, 5.55288, 9.32646, 19.5483,
27.6226, 40.3695, 50.6452, 67.0622, 81.9948,
102.578, 121.64, 144.626, 170.994, 198.418,
225.766, 259.499, 292.716, 328.8, 365.317, 402.559}

なお，ここでは詳細について説明はしませんが，データの行と列を入れ換える命令「Transpose」を使うことで，縦に延びるデータを横に延びるデータに，横に延びる

In[16]:= **Transpose[First[縦形式の生データ]]**

Out[16]= {{実施日, 1., 2., 3., 4., 5., 6., 7., 8., 9., 10.,
11., 12., 13., 14., 15., 16., 17., 18., 19., 20.},
{測定値, 4.77725, 5.55288, 9.32646, 19.5483, 27.6226,
40.3695, 50.6452, 67.0622, 81.9948, 102.578,
121.64, 144.626, 170.994, 198.418, 225.766,
259.499, 292.716, 328.8, 365.317, 402.559}}

In[17]:= **First[Transpose[First[縦形式の生データ]]]**

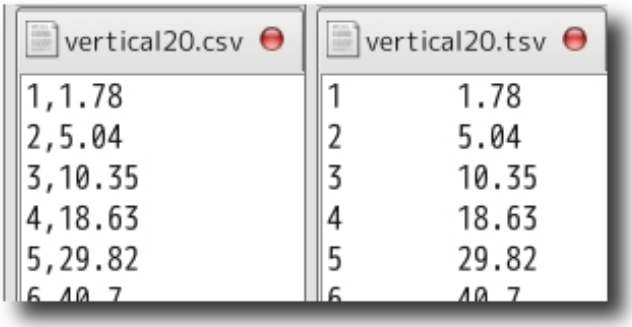
Out[17]= {実施日, 1., 2., 3., 4., 5., 6., 7., 8., 9., 10.,
11., 12., 13., 14., 15., 16., 17., 18., 19., 20.}

【パワフルなImport】*Mathematica*に表計算ソフトの情報を読み込むために利用したImportという命令は，表計算ソフトだけでなく様々なデータ形式のファイルを読み込むことが出来ます．扱える形式は「$ImportFormats」という命令で確認できま

■ **実験データを直接読み込んでみよう**

*Mathematica*を使いはじめた理由によっては，CSV（カンマ区切り）やTSV（タブ区切り）形式で保存しているかもしれません．でも安心してください．表計算ソフトで保

ここでは実験データが次のようにCSV形式とTSV形式で保存されているとしましょう．下記の説明ではファイル名を「vertical20.csv」と「vertical20.tsv」としています



実験データ：CSV形式とTSV形式

表計算ソフトのデータと同じくImportという命令に，次のように読み込みたいファイル名を文字列として指示して読み込みます．読み込んだデータには名前を付けておきましょう．ファイル名を入力するのが面倒な場合は，前述の通り，メニューの「挿入

```
In[19]:= CSVのデータ = Import["vertical20.csv"]

Out[19]= {{1, 1.78}, {2, 5.04}, {3, 10.35}, {4, 18.63},
          {5, 29.82}, {6, 40.7}, {7, 51.98}, {8, 65.53},
          {9, 83.06}, {10, 102.35}, {11, 124.69}, {12, 147.71},
          {13, 171.23}, {14, 200.74}, {15, 225.5}, {16, 259.76},
          {17, 289.63}, {18, 324.55}, {19, 364.03}, {20, 402.24}}

In[20]:= TSVのデータ = Import["vertical20.tsv"]

Out[20]= {{1, 1.78}, {2, 5.04}, {3, 10.35}, {4, 18.63},
          {5, 29.82}, {6, 40.7}, {7, 51.98}, {8, 65.53},
          {9, 83.06}, {10, 102.35}, {11, 124.69}, {12, 147.71},
          {13, 171.23}, {14, 200.74}, {15, 225.5}, {16, 259.76},
          {17, 289.63}, {18, 324.55}, {19, 364.03}, {20, 402.24}}
```

*Mathematica*に読み込まれたデータは，表計算ソフトのファイル形式でも，CSVやTSV形式でも同じように扱うことが出来ます．従って，同じようにブラケット2つの組を使えば好きなところを切り出すことが出来ます．例えば，10番目や12番目のデータ

```
In[21]:= CSVのデータ[[10]]

Out[21]= {10, 102.35}
```

```
In[22]:= CSVのデータ[[12]]
Out[22]= {12, 147.71}
```

実施日に関するデータだけ，測定値に関するデータだけを取り出したい場合にも，表計算ソフトの場合と同じく「All」を使うことで次のように取り出すことが出来ま

```
In[23]:= CSVのデータ[[All, 1]]
Out[23]= {1, 2, 3, 4, 5, 6, 7, 8, 9, 10,
          11, 12, 13, 14, 15, 16, 17, 18, 19, 20}

In[24]:= CSVのデータ[[All, 2]]
Out[24]= {1.78, 5.04, 10.35, 18.63, 29.82, 40.7, 51.98, 65.53,
          83.06, 102.35, 124.69, 147.71, 171.23, 200.74,
          225.5, 259.76, 289.63, 324.55, 364.03, 402.24}
```

【数の表現の多様性】実験データにおける数の表現は，数学的な表現だけでなく，「E」を使ったものなど様々なものがあります．*Mathematica*のImportは実験データなどに含まれる多様な数の表現を自動認識するので，特に意識することなく

- **面倒だから*Mathematica*の中でデータを作ってしまおう**

表計算ソフトは使っていないけれど，データ処理を行いたい場合もあります．いわゆるシミュレーションです．*Mathematica*にそのようなデータを作らせるためには，双子

今回，取り扱うデータの種類は2次元データになります．従って，それぞれのデータは対で表現されます．例えば，双子素数の際に使用した方法を模して作るとすれば，次

```
In[25]:= Table[{i, Prime[i]}, {i, 1, 20}]
Out[25]= {{1, 2}, {2, 3}, {3, 5}, {4, 7}, {5, 11},
          {6, 13}, {7, 17}, {8, 19}, {9, 23}, {10, 29},
          {11, 31}, {12, 37}, {13, 41}, {14, 43}, {15, 47},
          {16, 53}, {17, 59}, {18, 61}, {19, 67}, {20, 71}}
```

前項などで扱った形式のデータ，すなわち，x座標が$1, 2, 3, \ldots$となるにつれて，y座標がx^2にランダムな誤差を加えたものをTableで作成するには次のようにします．

```
In[26]:= Table[{i, i^2 + RandomReal[{0, 5}]}, {i, 1, 20}]

Out[26]= {{1, 3.44449}, {2, 8.30703}, {3, 9.16048}, {4, 18.7909},
          {5, 27.4718}, {6, 36.6054}, {7, 53.8471}, {8, 66.1074},
          {9, 81.3053}, {10, 104.886}, {11, 125.119},
          {12, 146.611}, {13, 170.026}, {14, 196.671},
          {15, 225.73}, {16, 258.882}, {17, 290.286},
          {18, 327.023}, {19, 362.367}, {20, 400.484}}
```

もしくは，より分かり易く変数にxを使うならば次のようになります．

```
Table[{x, x^2 + RandomReal[{0, 5}]}, {x, 1, 20}]
```

表計算ソフトから読み込んだデータと同じではつまらないので，ここでは多少変更してデータを生成して，結果を「生成データ」として*Mathematica*に覚えさせておきま

```
In[27]:= 生成データ = Table[{x, -x^2 + 300 + RandomReal[{0, 50}]},
           {x, 1, 20}];
```

このデータの次元を確認すると，前項などで読み込んだデータと同じ次元になってい

```
In[28]:= Dimensions[生成データ]

Out[28]= {20, 2}
```

もちろん，x軸だけの取り出しや，y軸だけの取り出し，データの行と列を入れ換えることも同様に可能です．

```
In[29]:= 生成データ[[All, 1]]

Out[29]= {1, 2, 3, 4, 5, 6, 7, 8, 9, 10,
          11, 12, 13, 14, 15, 16, 17, 18, 19, 20}
```

```
In[30]:= 生成データ[[All, 2]]

Out[30]= {335.14, 318.404, 292.545, 319.12, 315.785,
          303.398, 282.894, 259.457, 251.296, 211.562,
          189.721, 181.96, 150.144, 127.566, 75.2987,
          92.8222, 17.4194, -19.1319, -55.5681, -74.5124}
```

In[31]:=
```
Transpose[生成データ]
```

Out[31]=
```
{{1, 2, 3, 4, 5, 6, 7, 8, 9, 10,
  11, 12, 13, 14, 15, 16, 17, 18, 19, 20},
 {335.14, 318.404, 292.545, 319.12, 315.785,
  303.398, 282.894, 259.457, 251.296, 211.562,
  189.721, 181.96, 150.144, 127.566, 75.2987,
  92.8222, 17.4194, -19.1319, -55.5681, -74.5124}}
```

データを分析して分かり易く可視化してみよう

■ データをとりあえず表形式で表示してみよう

*Mathematica*内でTableを使って生成したデータはリストのリスト，すなわち2次元リスト（2次元配列，2次元データ）になっていますので，リスト（グループ）について

In[32]:=
```
Grid[生成データ, Frame → All]
```

Out[32]=

1	335.14
2	318.404
3	292.545
4	319.12
5	315.785
6	303.398
7	282.894
8	259.457
9	251.296
10	211.562
11	189.721
12	181.96
13	150.144
14	127.566
15	75.2987
16	92.8222
17	17.4194
18	-19.1319
19	-55.5681
20	-74.5124

今回のデータは高々20組しか含まれていませんが，実際には大量のデータを扱うことが多いでしょう．そのような場合，データの1番目から4番目だけ表示したい，ということもあります．特定の□番目だけの取り出しであれば，表計算ソフトからのデータ読み込みの際に学んだ，リストの直後にブラケット2つの組（「[[」と「]]」）で数

既に学んだ方法と関連する方法として，要素の削除に使ったDropとは反対に「リストの最初の要素から指定した個数分だけ取り出しなさい」という意味を持つ「Take」を使う方法があります．使いかたはDropと全く同じです．

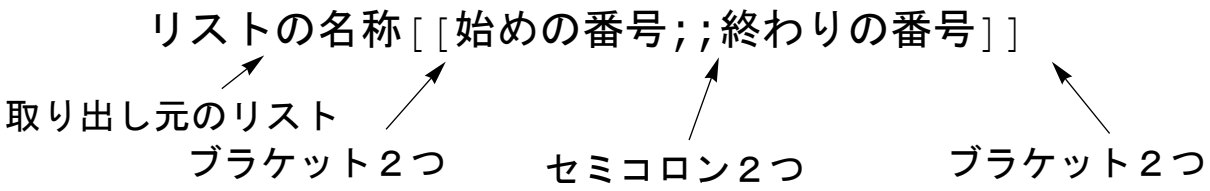
In[33]:=
```
Take[生成データ, 4]
```

Out[33]=
```
{{1, 335.14}, {2, 318.404}, {3, 292.545}, {4, 319.12}}
```

しかしながら，*Mathematica* 6にはより便利な方法が用意されています．それは，リストの直後にブラケット2つの組（「[[」と「]]」）の間に，セミコロン2つ（;;）の左右に1番目から4番目と書く方法です．次のように指示を出すことで，データの1番目か

In[34]:=
```
生成データ[[1 ;; 4]]
```

Out[34]=
```
{{1, 335.14}, {2, 318.404}, {3, 292.545}, {4, 319.12}}
```



リスト操作：データの一部分だけを取り出す

従って，生成データの一部分だけ（例えば，1番目から4番目，17番目から20番目など）を取り出して，それを表形式で表示するには次のように指示を出すことになりま

In[35]:=
```
Grid[生成データ[[1 ;; 4]], Frame → All]
```

Out[35]=

1	335.14
2	318.404
3	292.545
4	319.12

In[36]:=
```
Grid[生成データ[[17 ;; 20]], Frame → All]
```

Out[36]=

17	17.4194
18	−19.1319
19	−55.5681
20	−74.5124

データの行と列を入れ換えるTransposeを使うと，横に広がる表形式で表示させることも出来ます．この場合，データの数が大いと右側の方にスクロールしないと全体像が見えなくなってしまいます．実際，「個々の要素を非常に小さく表示しなさい」と

In[37]:=
```
Grid[Transpose[生成データ[[5 ;; 16]]], Frame → All,
 ItemStyle → Tiny]
```

Out[37]=

5	6	7	8	9	10	11	12	13	14	15	16
315.785	303.398	282.894	259.457	251.296	211.562	189.721	181.96	150.144	127.566	75.2987	92.8222

なお，最初に読み込んだような表計算ソフトのデータの場合は，次のようになります．最初のデータ対が項目名になっているので，そのままで非常に見やすいものにな

In[38]:=
```
Grid[最初のシート[[1 ;; 11]], Frame → All]
```

Out[38]=

実施日	測定値
1.	4.77725
2.	5.55288
3.	9.32646
4.	19.5483
5.	27.6226
6.	40.3695
7.	50.6452
8.	67.0622
9.	81.9948
10.	102.578

同様に，データの行と列を入れ換えるTransposeを使うと，横に広がる表形式で表示させることも出来ます．この場合，データの数が大いと右側の方にスクロールしない

In[39]:=
```
Grid[Transpose[最初のシート[[1 ;; 11]]], Frame → All,
 ItemStyle → Tiny]
```

Out[39]=

実施日	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.
測定値	4.77725	5.55288	9.32646	19.5483	27.6226	40.3695	50.6452	67.0622	81.9948	102.578

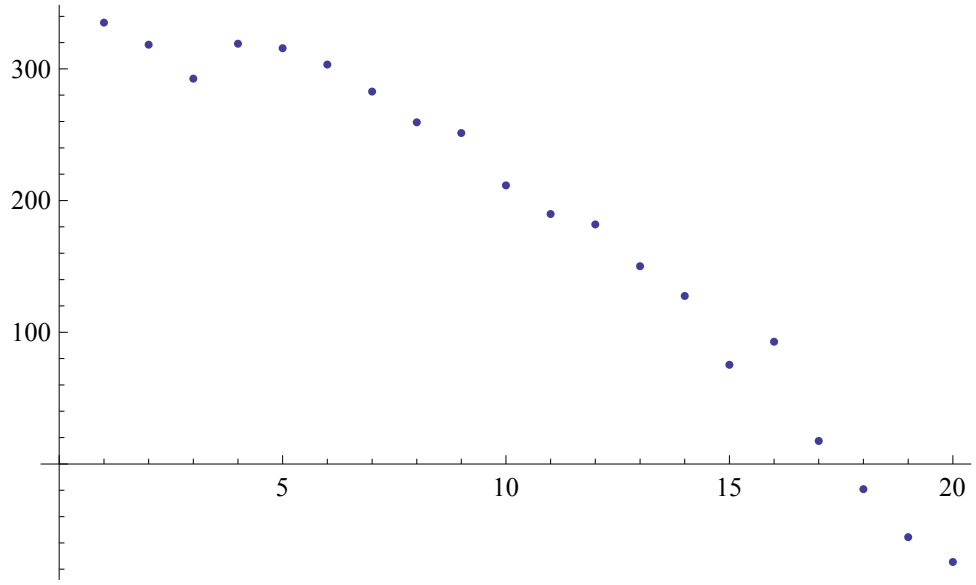
【Takeによる部分取り出し】*Mathematica*では同じ操作を様々な方法で行えると，何度も説明しています．「リスト[[n;;m]]」という形式での部分取り出しも，違う方法で行うことも可能です．例えば，「Take[リスト,{n,m}]」や「リスト

■ データを点グラフで表示してみよう

今度はデータをグラフで表示してみましょう．グラフで表示させるには，「次のデータを点グラフで表示しなさい」という意味の「ListPlot」という命令を使いま

In[40]:= **ListPlot[生成データ]**

Out[40]=



ListPlot[データのリスト]

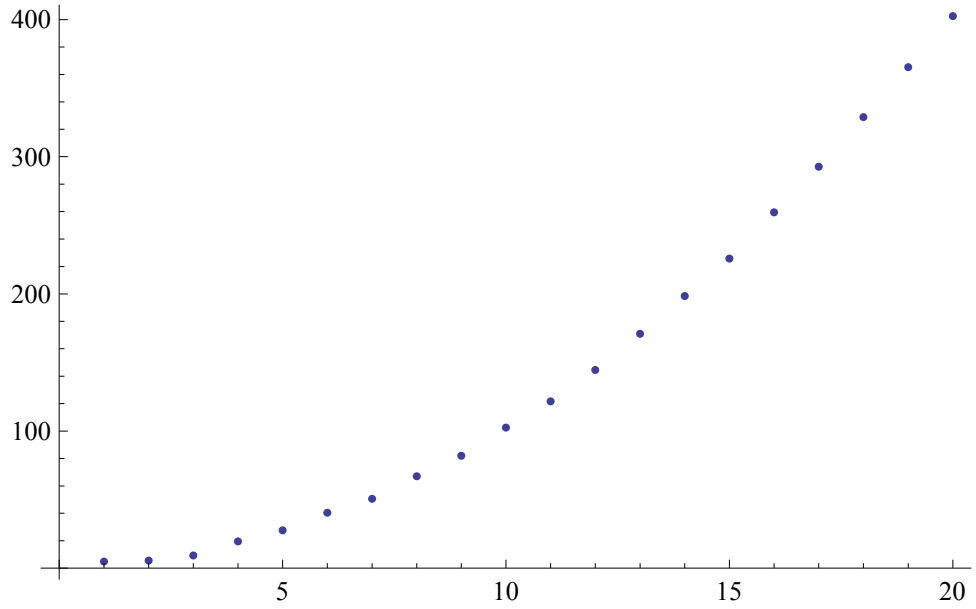
表示させたいデータのリスト　　必要に応じてオプションが付く

データの可視化：データを点グラフで表示する

表計算ソフトから読み込んだ最初のシートのデータには，項目名である文字列も含まれていますが，*Mathematica*は「文字列は可視化すべきデータではない」と判断して表

In[41]:= **ListPlot[最初のシート]**

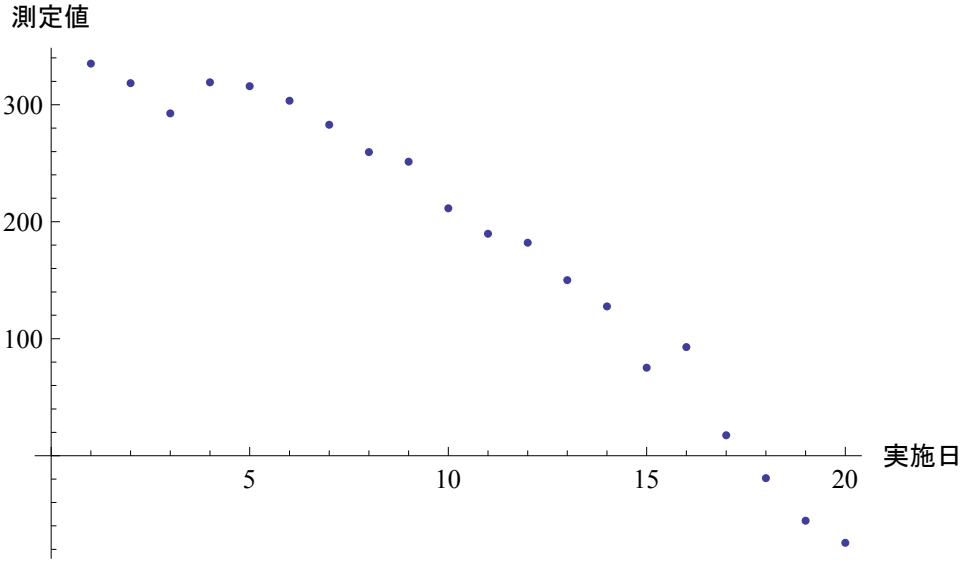
Out[41]=



グラフの軸に名前を付けるには，「各軸にラベルを付けなさい」という意味のオプシ

ョン「AxesLabel」を使います．このオプションには，次の例のようにリストでx軸に付けるラベルとy軸に付けるラベルを文字列（二重引用符で囲む）で指定します．

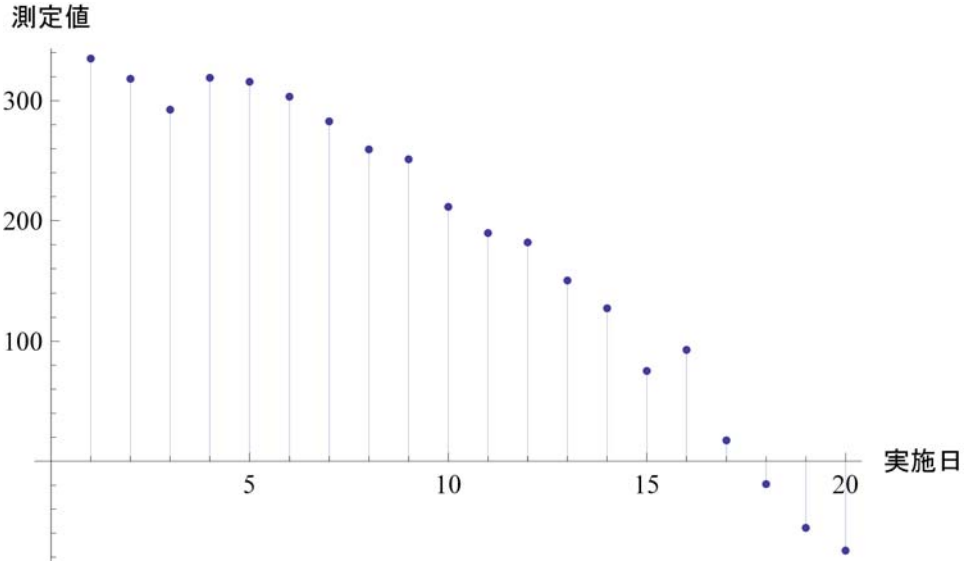
```
In[42]:= ListPlot[生成データ, AxesLabel → {"実施日", "測定値"}]
```



Out[42]=

点グラフだけでは若干見づらいので，「点から軸までを線で結びなさい（軸までを線で埋めなさい）」という意味のオプション「Filling → Axis」を使うことで，より

```
In[43]:= ListPlot[生成データ, AxesLabel → {"実施日", "測定値"},
          Filling → Axis]
```



Out[43]=

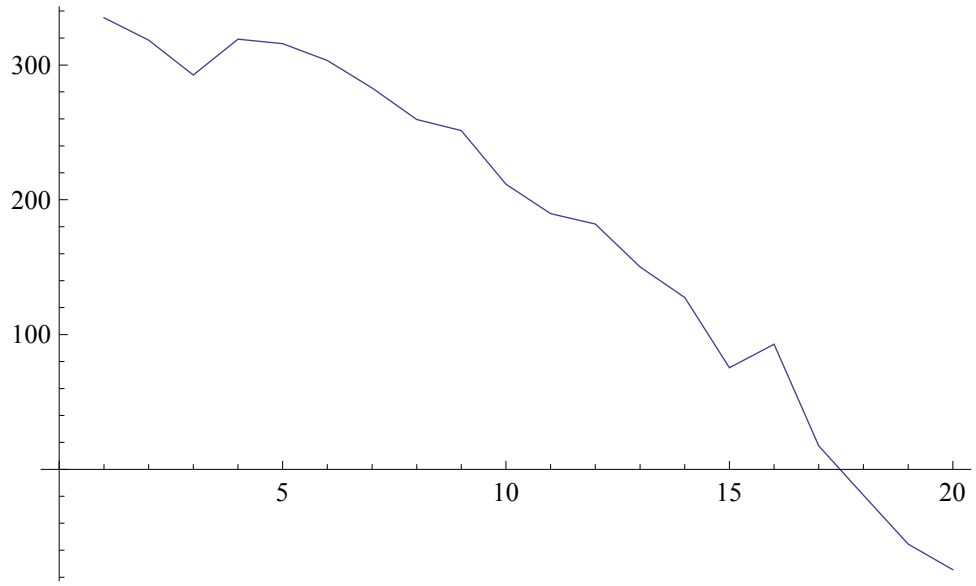
【グラフのオプション】*Mathematica*のグラフィックスに関連する命令には非常に多くのオプションが用意されています．余りにも多いため，本書で全てを扱ってはいません．細かい設定に興味を持った方は，各グラフの命令（今回の例で言え

■ **データを折れ線グラフで表示してみよう**

今度はデータを折れ線グラフで表示してみましょう．折れ線グラフで表示させるには，「次のデータを折れ線グラフで表示しなさい」という意味の

```
In[44]:= ListLinePlot[生成データ]
```

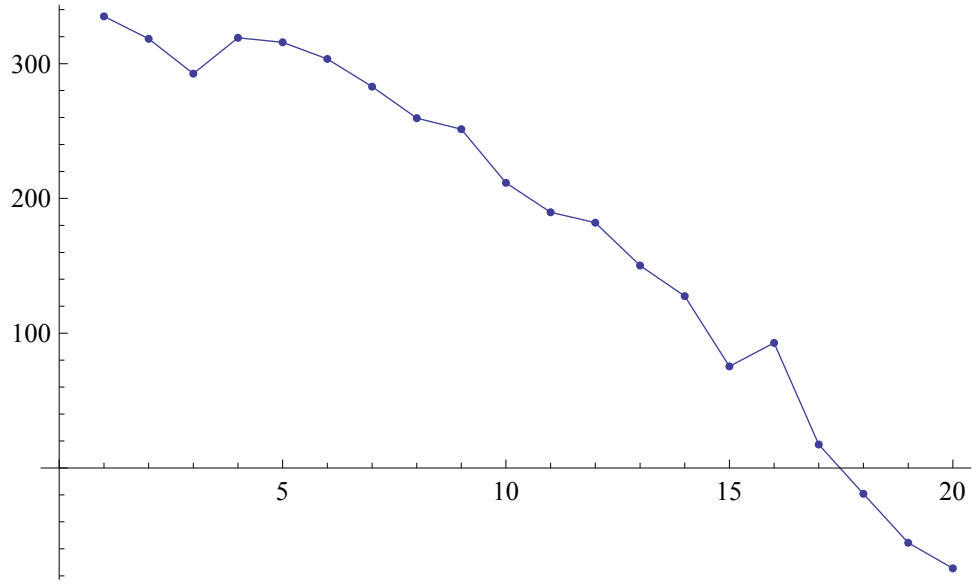
Out[44]=



しかし，単純な折れ線では線分を確認するのが難しいので，次のように「全ての線分の区切りを表示しなさい」という意味のオプション「Mesh → All」を追加して使用

In[45]:= **ListLinePlot[生成データ, Mesh → All]**

Out[45]=



ListLinePlot[データのリスト, Mesh→All]

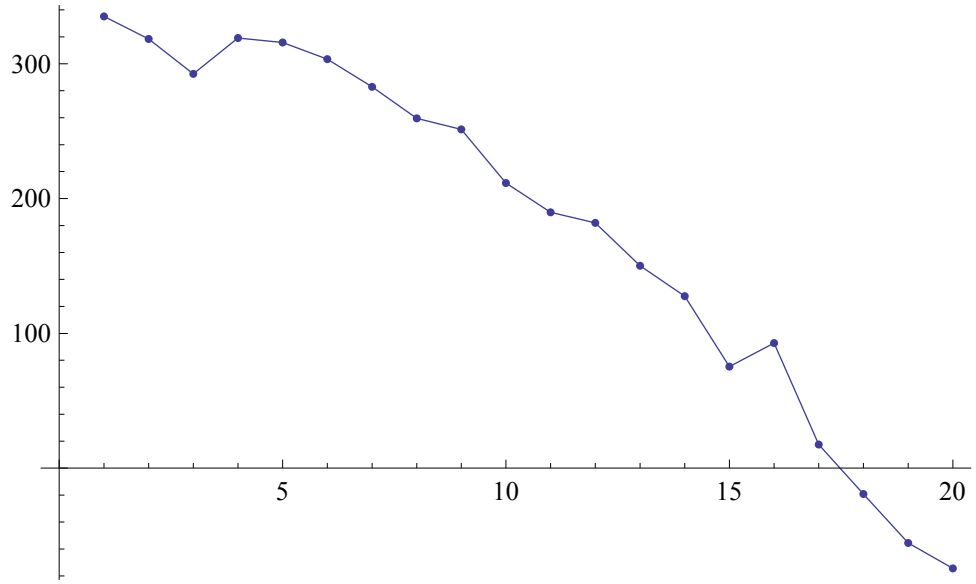
表示させたいデータのリスト　　線分を分かり易くするオプション

データの可視化：データを折れ線グラフで表示する

前項で使用したListPlotと今回のListLinePlotは非常に良く似た命令なので，次のように前項のListPlotにいくつかのオプションを組み合わせることでも同じグラフを描かせることが可能です．オプションの「Joined → True」は「点と点を線で

In[46]:= **ListPlot[生成データ, Joined → True, Mesh → All]**

Out[46]=

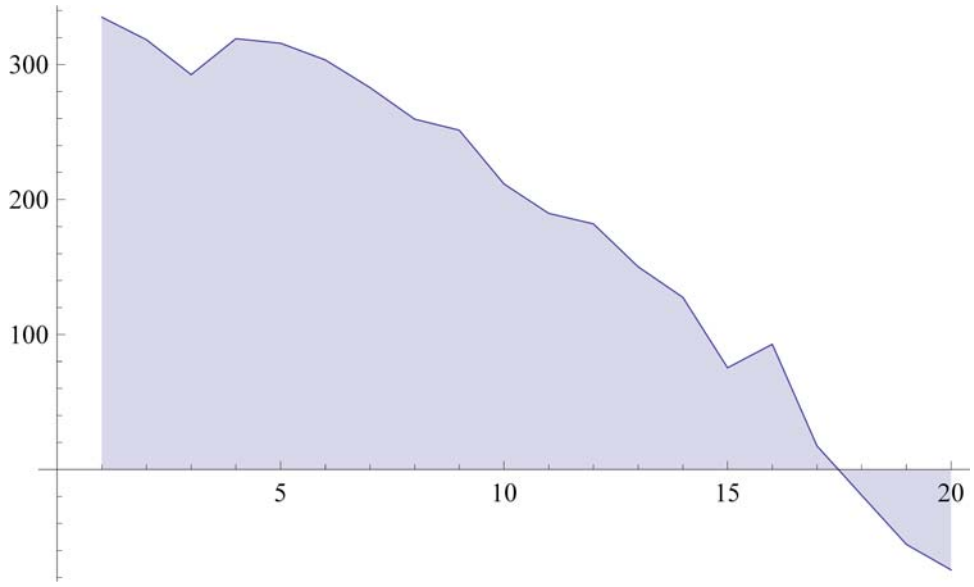


前項と同じく，「線から軸までを色を塗って埋めなさい」という意味のオプション

「Filling → Axis」を使うことで，より見やすくすることも出来ます．

In[47]:=
```
ListLinePlot[生成データ, Filling → Axis]
```

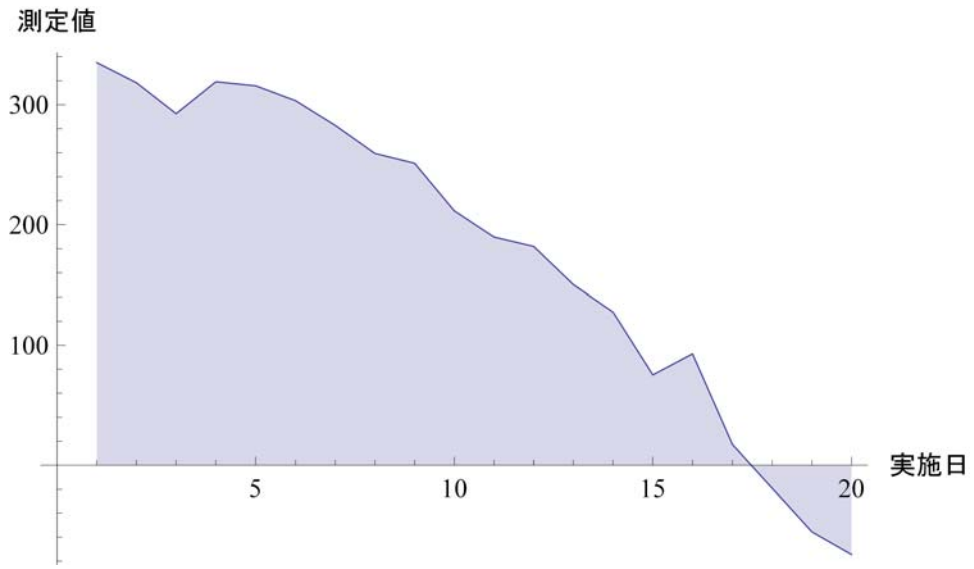
Out[47]=



なお，各軸にラベルを付けるには前項と同じくAxesLabelを使います．

In[48]:=
```
ListLinePlot[生成データ, Filling → Axis,
 AxesLabel → {"実施日", "測定値"}]
```

Out[48]=

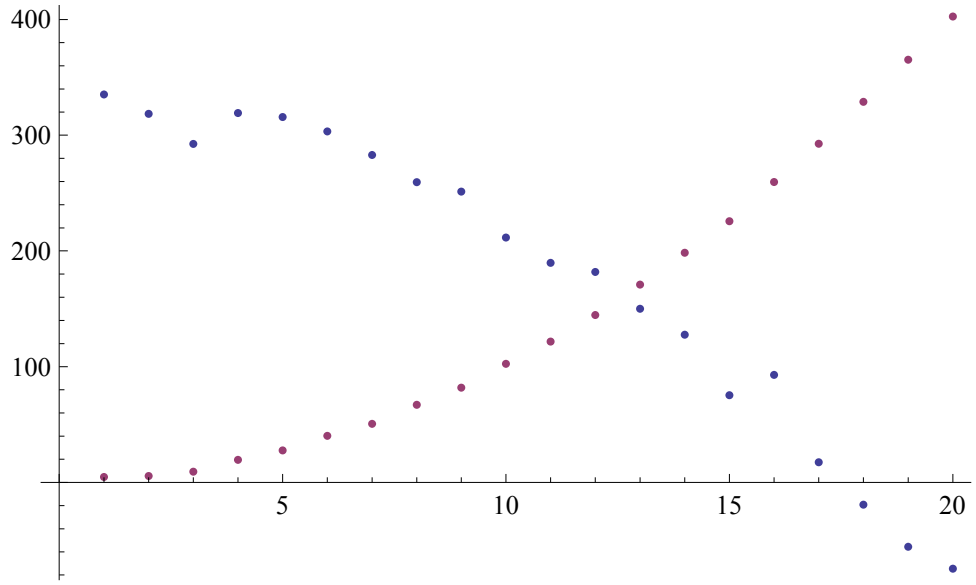


■ **複数のデータを同時にグラフに表示してみよう**

複数のデータ列がある場合は，同じグラフに表示させて比較検討を行いたいと思います．表計算ソフトであれば，グラフ描画の範囲指定で複数のデータ列を選択しておけば，そのようなグラフを描けます．*Mathematica*で複数のデータ列をプロットするには，他の命令と同じくリストを使って複数のデータ列を指定します．折れ線グラフに

In[49]:=
```
ListPlot[{生成データ, 最初のシート}]
```

Out[49]=



ListPlot[{データ1のリスト,データ2のリスト,...}]

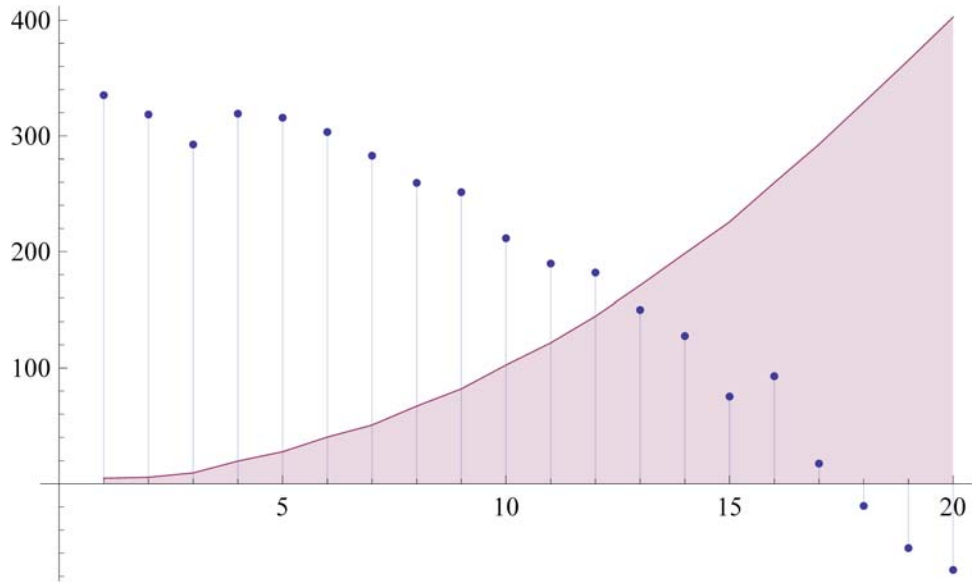
何組のデータ列もリストで与えれば描画される

データの可視化：複数のデータ列を点グラフで表示する

複数のデータ列を指定した場合，それぞれについて点グラフにするか折れ線グラフにするかを，オプションJoinedで指示することが出来ます．次の命令では，生成データ

In[50]:=
```
ListPlot[{生成データ, 最初のシート}, Filling → Axis,
 Joined → {False, True}]
```

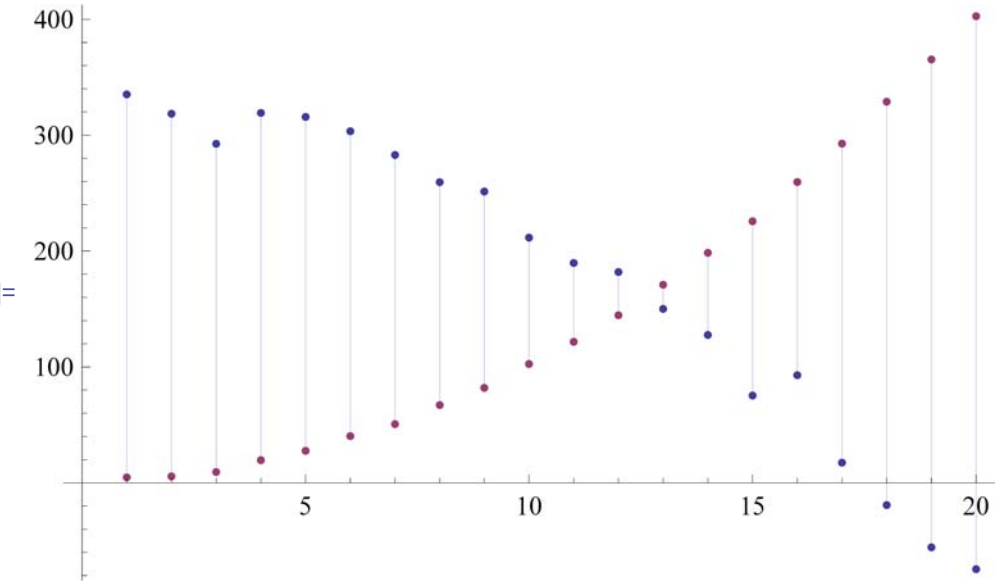
Out[50]=



「点から軸までを線で結びなさい（軸までを線で埋めなさい）」という補足指示に使用したオプションFillingを「Filling → {1 → {2}}」と指定することで，「1番目のデータから2番目のデータまでを線で結びなさい」という指示を出すことも出来ます．

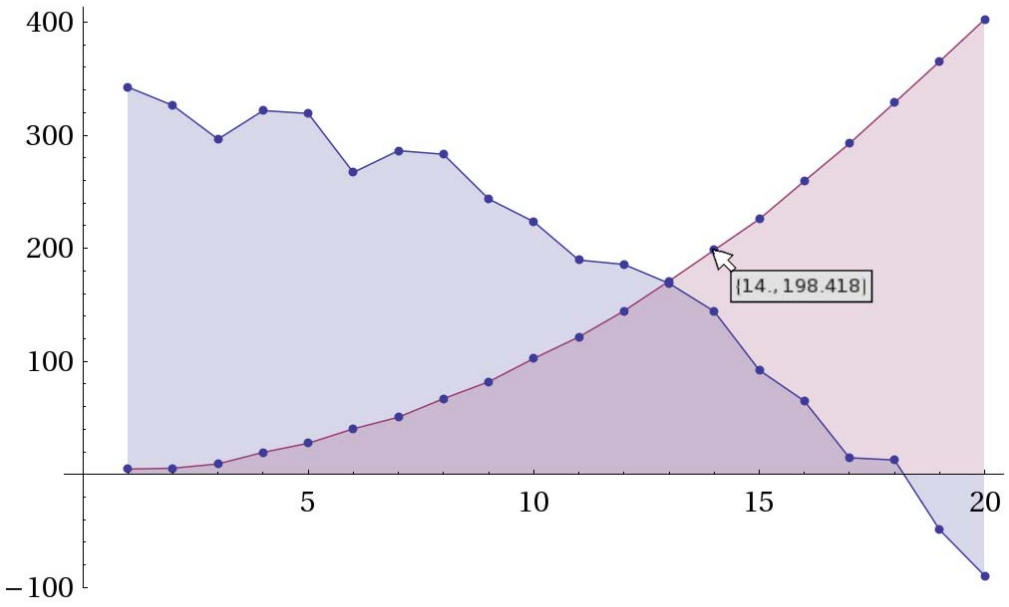
In[51]:=
```
ListPlot[{生成データ, 最初のシート}, Filling → {1 → {2}}]
```

Out[51]=



このようにして複数のデータ列を扱ってグラフによる比較検討をする場合，グラフ上の点の座標値を正確に知りたくなることがあります．*Mathematica*には，「グラフ上の点にマウスを重ねたら，データの詳細をツールチップで表示しなさい」という意味の

```
ListLinePlot[Tooltip[{生成データ, 最初のシート}],
 Mesh → All, Filling → Axis]
```



【全てはリスト】*Mathematica*の基本はリストです．複数のものを扱うときは必ずリストが出てきます．この基本を理解していれば，複数のデータ列をグラフに表示するときのリスト指定，オプションによる点グラフと折れ線グラフの切替えのリスト指定，などが自然な方法だと感じられると思います．

回帰分析やフィッティングで関係式を求めてみよう

- **1次式の回帰直線を求めてみよう**

表計算ソフトでは統計的な処理を行うことが多いと思います．そこで，生成データの回帰直線を求めてみたいと思います．操作は非常に簡単で，*Mathematica*に対して「最小二乗法で次のデータの回帰直線を求めなさい」という意味を持つ命令「Fit」で次

```
In[52]:= 回帰直線 = Fit[生成データ, {1, x}, x]
Out[52]= 408.561 − 21.8852 x
```

Fit[データのリスト, {1,x}, x]

1次式なので0次と1次の項のリスト

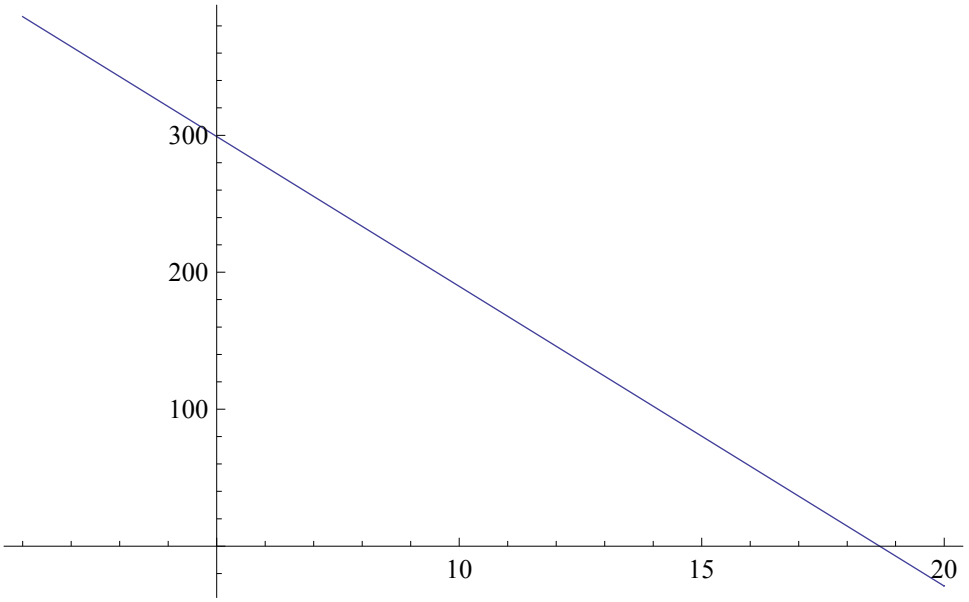
回帰直線の目的変数を指定（何でも良い）

フィッティング：データの回帰直線を計算するには

求まった回帰直線のグラフを描かせるには，「指定した変数の範囲で，次の関数のグラフを描きなさい」という意味の命令「Plot」を使います．実際に，データ列で与えられている目的変数xの範囲「$1 \le x \le 20$」で描かせてみましょう．

In[53]:= **Plot[回帰直線, {x, 1, 20}]**

Out[53]=



Plot[関数, {x, 1, 20}]

グラフにしたい関数

関数の変数（この変数についての関数のグラフ）

描画範囲の始点

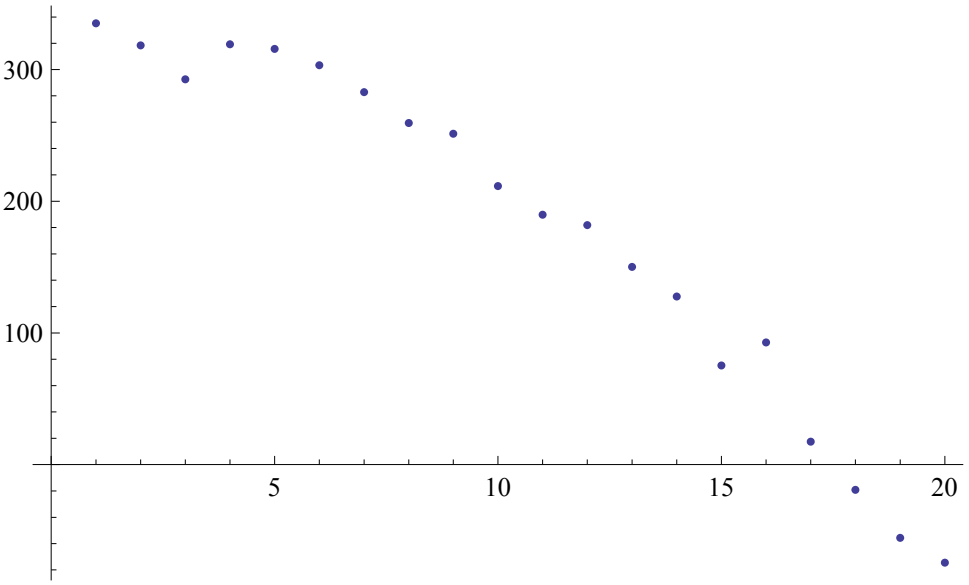
描画範囲の終点

関数のグラフ描画：陽関数表示の2次元グラフを描く

元のデータ列と比べるために，ListPlotによる点グラフを「グラフ1」というシンボルに覚えさせておきます．*Mathematica*では，数や文字列だけでなく，グラフィックス

In[54]:= **グラフ1 = ListPlot[生成データ]**

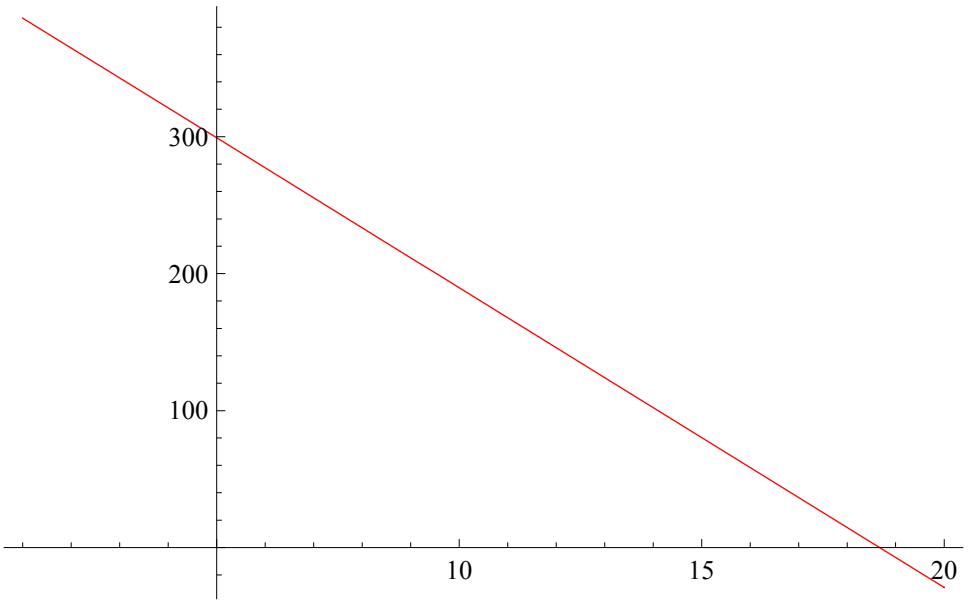
Out[54]=



同様に回帰直線のグラフも「グラフ2」というシンボルに覚えさせておきます．このとき，元のグラフと異なる色がわかりやすいので，オプション「PlotStyle → Red」により「グラフを描くときに，指定の色（赤）を使いなさい」と指示しておきます．色としてはRedの他，青（Blue），緑（Green）なども使えます．

```
In[55]:= グラフ2 = Plot[回帰直線, {x, 1, 20}, PlotStyle → Red]
```

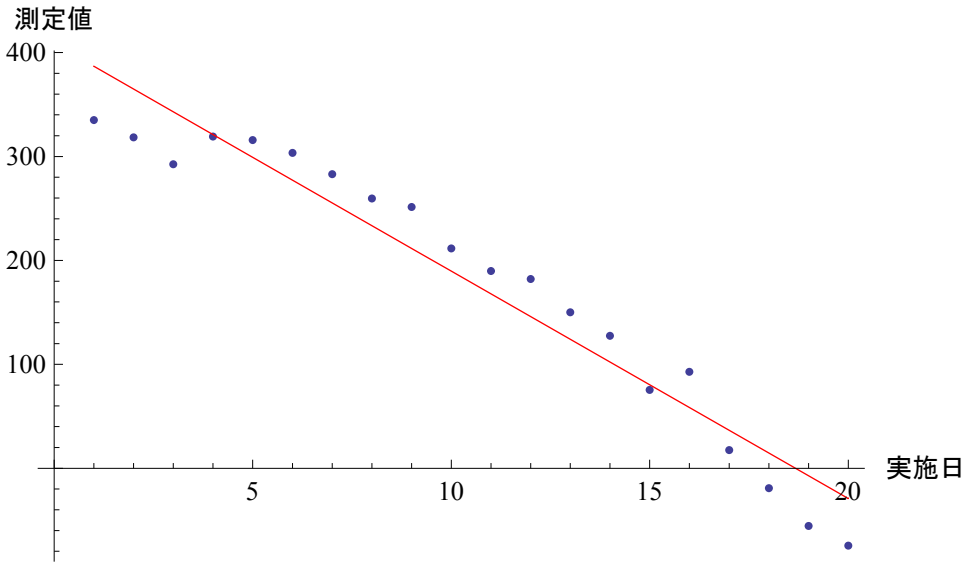
Out[55]=

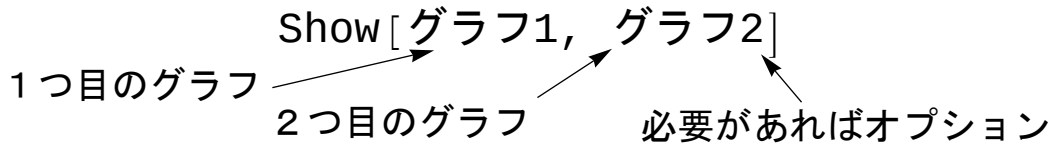


複数のグラフを重ねて表示するには，「複数のグラフを同時に表示しなさい」という命令「Show」を使います．Showには既に覚えさせているグラフを複数個指定するだけで，それらのグラフが重なった表示されます．実際に，元のデータ列の点グラフと

In[56]:=
```
Show[グラフ1, グラフ2, AxesLabel -> {"実施日", "測定値"}]
```

Out[56]=





関数のグラフ描画：複数のグラフを重ね合わす

なお，わざわざシンボルに覚えさせるのが面倒な場合や，覚えさせる必要がない場合は，次のように直接ShowにListPlotやPlotなどの命令を組み合わせることも出来ま

```
Show[ListPlot[生成データ],
 Plot[回帰直線, {x, 1, 20}, PlotStyle → Red],
 AxesLabel -> {"実施日", "測定値"}]
```

【グラフの重ね合わせ】*Mathematica* 6以前では，グラフの重ね合わせの際に特殊なオプション指定（DisplayFunction）をする必要がありましたが，*Mathematica* 6からグラフィックスも通常の数式と同じように扱えるため，非常にシンプルな命令だけで

求めた回帰直線の式を使って，回帰直線上のデータ列を作ることも可能です．リストを作ることになるので，双子素数の探索などでも使ったTableを利用します．実際に

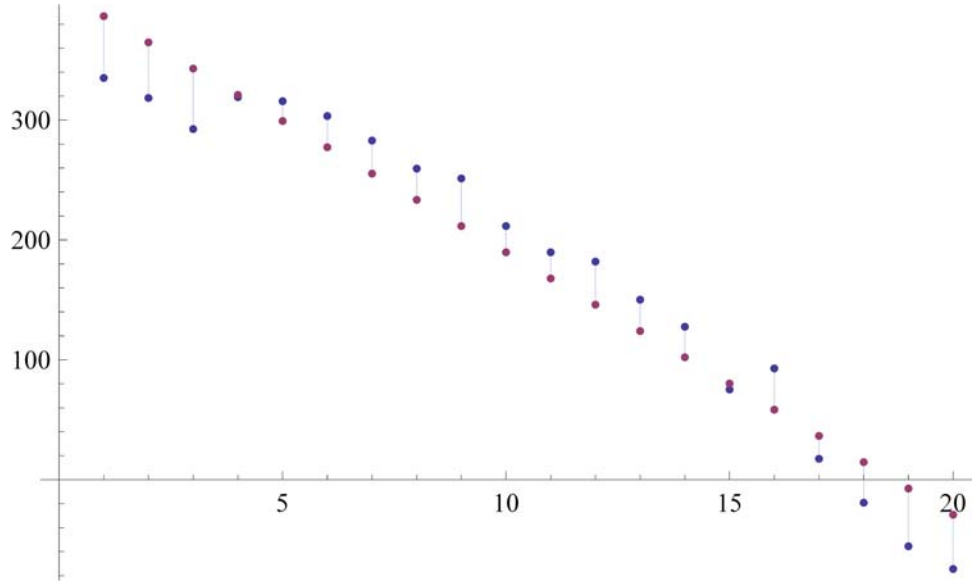
In[57]:=
```
回帰データ = Table[{x, 回帰直線}, {x, 1, 20}]
```

Out[57]=
```
{{1, 386.676}, {2, 364.79}, {3, 342.905}, {4, 321.02},
 {5, 299.135}, {6, 277.249}, {7, 255.364}, {8, 233.479},
 {9, 211.594}, {10, 189.709}, {11, 167.823},
 {12, 145.938}, {13, 124.053}, {14, 102.168},
 {15, 80.2824}, {16, 58.3972}, {17, 36.5119},
 {18, 14.6267}, {19, -7.25855}, {20, -29.1438}}
```

結果，元のデータ列「生成データ」と回帰直線のデータ列「回帰データ」が準備できたので，これらを複数のデータ列をグラフ表示することで，次のようによりわかりや

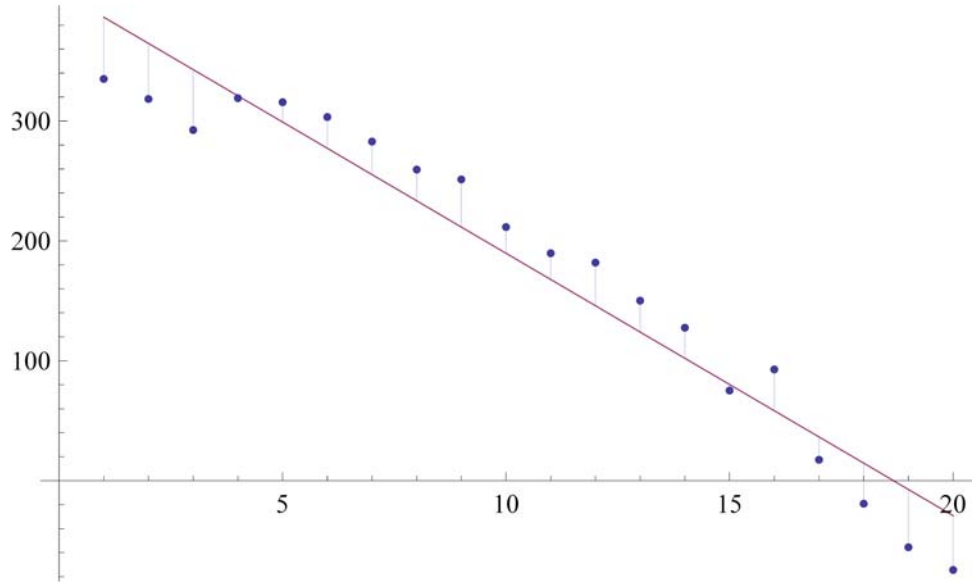
In[58]:=
```
ListPlot[{生成データ, 回帰データ}, Filling → {1 → {2}}]
```

Out[58]=



In[59]:=
```
ListPlot[{生成データ, 回帰データ}, Joined → {False, True},
 Filling → {1 → {2}}]
```

Out[59]=



■ **より高次な回帰曲線を求めてみよう**

「最小二乗法で次のデータの回帰直線を求めなさい」という命令「Fit」では，2次以上の項も含めて指示することで，任意の次数の曲線を使ってフィッティングを行わせることが可能です．下記の例では，結果のグラフを書くために，求めた回帰曲線を

In[60]:=
```
回帰曲線 = Fit[生成データ, {1, x, x^2}, x]
```

Out[60]=
$$329.274 - 0.261614\,x - 1.0297\,x^2$$

```
Fit[データのリスト, {1,x,x^2,...}, x]
```
必要な次数までの項のリスト

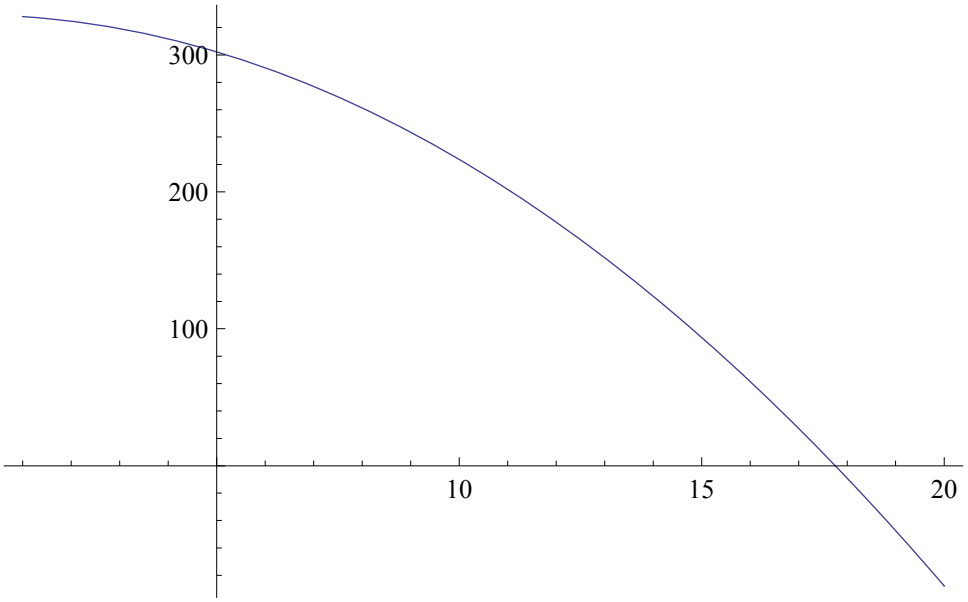
回帰直線の目的変数を指定（何でも良い）

フィッティング：データの回帰曲線を計算するには

求まった回帰曲線は，前項と同じくPlotを使うことでグラフを描くことが出来ます．実際に，データ列で与えられている目的変数xの範囲「$1 \le x \le 20$」で描かせてみましょう．

In[61]:=
```
Plot[回帰曲線, {x, 1, 20}]
```

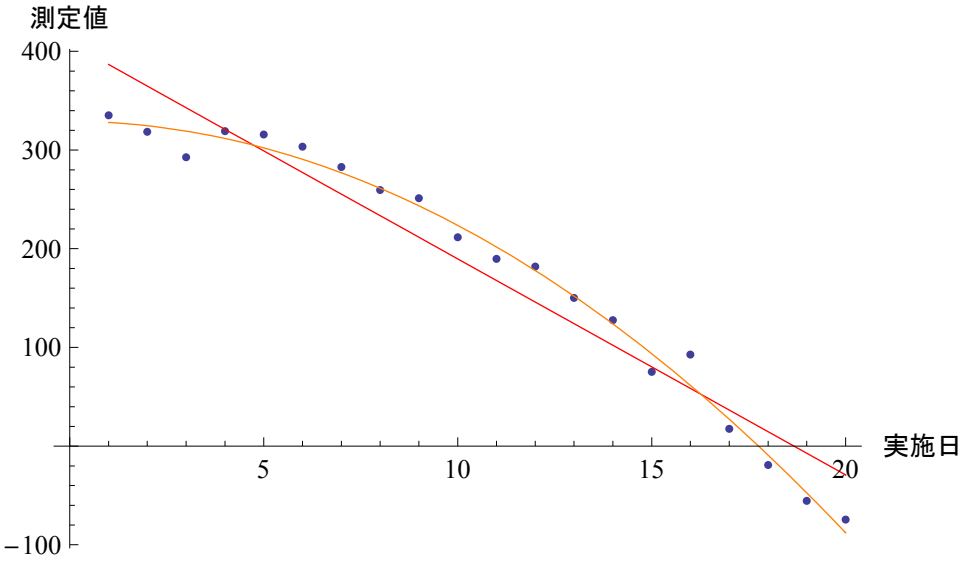
Out[61]=



元のデータ列とも，Showを使って比べてみましょう．既に覚えさせてある「グラフ1」と「グラフ2」に加えて，回帰曲線のグラフを重ね合わせています．やはり，元々

In[62]:=
```
Show[グラフ1, グラフ2,
 Plot[回帰曲線, {x, 1, 20}, PlotStyle → Orange],
 AxesLabel -> {"実施日", "測定値"}]
```

Out[62]=



同じ形のリストをたくさん作るのに適している組込み関数Tableを活用することで，幾つもの次数によるフィッティングを同時に作らせることも出来ます．下記では，生成データの回帰直線（1次多項式への最小二乗法による当てはめ）の他，2次から4次までの各多項式への最小二乗法による当てはめを計算させています．ここでの

In[63]:=
```
Table[Fit[生成データ, x^Range[0, n], x], {n, 1, 4}]
```

Out[63]=
$$\{408.561 - 21.8852\,x,\ 329.274 - 0.261614\,x - 1.0297\,x^2,\ 321.559 + 3.67497\,x - 1.4871\,x^2 + 0.0145208\,x^3,\ 325.363 + 0.69938\,x - 0.884973\,x^2 - 0.02933\,x^3 + 0.00104407\,x^4\}$$

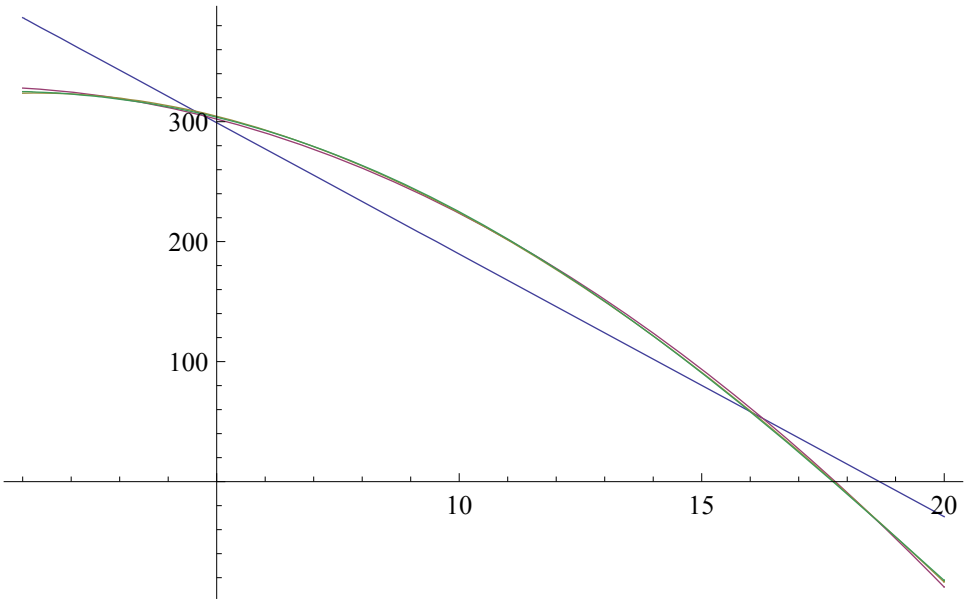
陽関数を描画するための命令であるPlotは，複数の関数をリストで指定することが出来ます．そのため，様々なフィッティングの結果を表示させようと，次のように指示

```
Plot[Table[Fit[生成データ, x^Range[0, n], x], {n, 1, 4}],
 {x, 1, 20}]
```

組込み関数のPlotは指示された関数を描画するために，その関数の評価を直前まで行いません．しかしながら，今回の場合は事前に評価しておいてもらわないと逆に困ります．このような場合は，「とにかく事前に評価しなさい」という意味を持つ命令

In[64]:=
```
回帰曲線など =
 Plot[Evaluate[Table[Fit[生成データ, x^Range[0, n], x],
    {n, 1, 4}]], {x, 1, 20}]
```

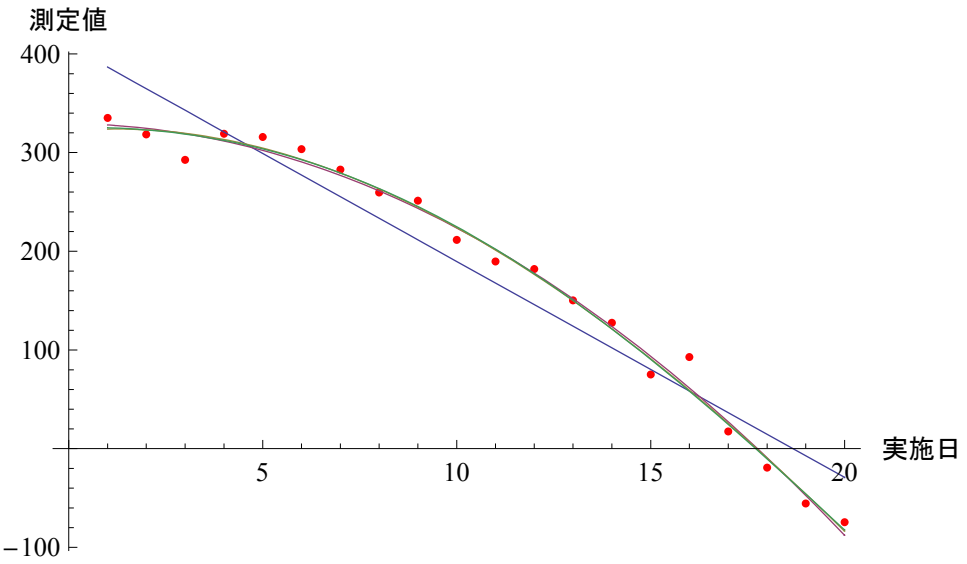
Out[64]=



これらの高次の項までを使ったフィッティングの結果を元のデータと併せて表示させてみましょう．元のデータ列を分かり易くするために，オプションで赤色を指定して

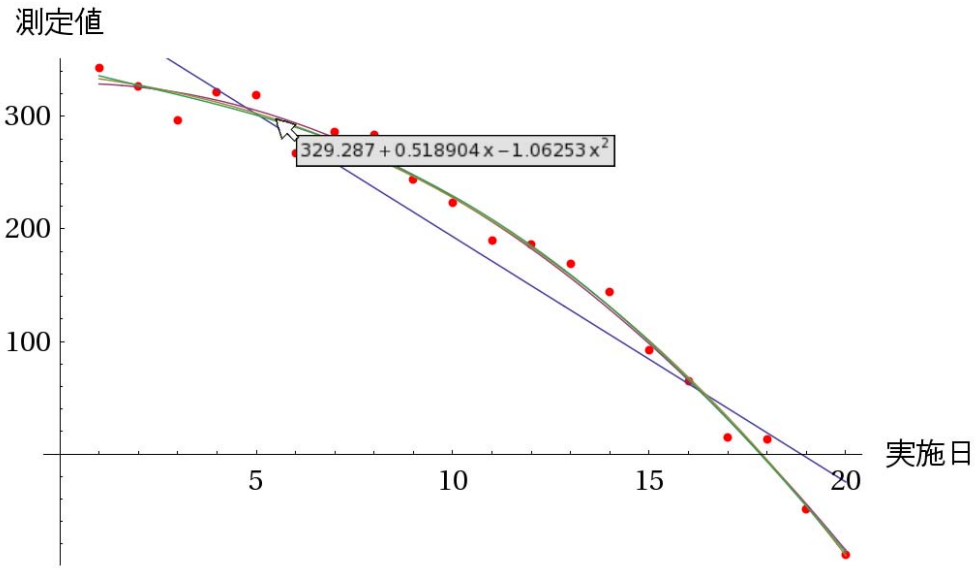
In[65]:=
```
Show[ListPlot[生成データ, PlotStyle → Red], 回帰曲線など,
 AxesLabel -> {"実施日", "測定値"}]
```

Out[65]=



以前に出てきたTooltipをうまく組み合わせて指示を行うことで，このような複雑な式でも，どの線が何次のフィッティングとなっているかをマウスで確認できるように

```
Show[ListPlot[Tooltip[生成データ], PlotStyle → Red],
 Plot[Evaluate[Tooltip[
    Table[Fit[生成データ, x^Range[0, n], x], {n, 1, 4}]
  ]], {x, 1, 20}], AxesLabel -> {"実施日", "測定値"}]
```

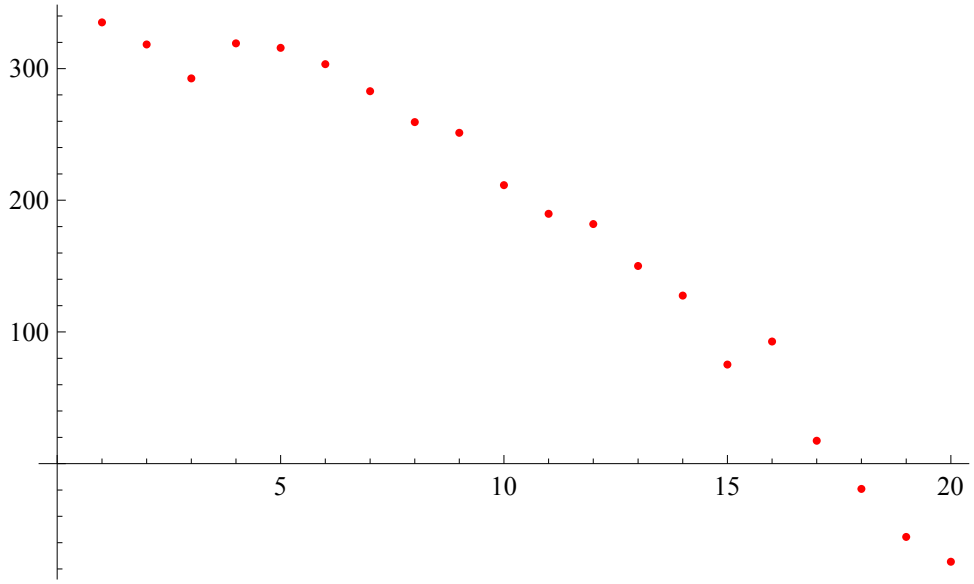


■ **複雑な非線形モデルを使った回帰曲線を求めてみよう**

前項までは多項式によるデータ列の当てはめを求めさせていましたが，より複雑な式，三角関数などの初等関数や未知のパラメータが初等関数の中に含まれるような非線形モデルを使ったフィッティングを行ってみましょう．ここでもフィッティングは

```
In[66]:= グラフ1 = ListPlot[生成データ, PlotStyle → Red]
```

Out[66]=



多項式への当てはめに使用した組込み関数Fitには，「x^n」形式だけでなく目的変数の任意の式を含めることが出来ます．例えば，三角関数も含めたい場合には，次のように「Sin[x]」や「Cos[x]」を加えることで，それらの係数パラメータを最小二乗法で求めさせることも出来ます．ここで，Sin[x]は「sin(x)」を，Cos[x]はcos(x)を表しており，　*Mathematica*　への指示における三角関数は，他の組込み関数と同じく最初の文字を大文字に，変数である「x」をブラケット（[]）で囲む必要があります．

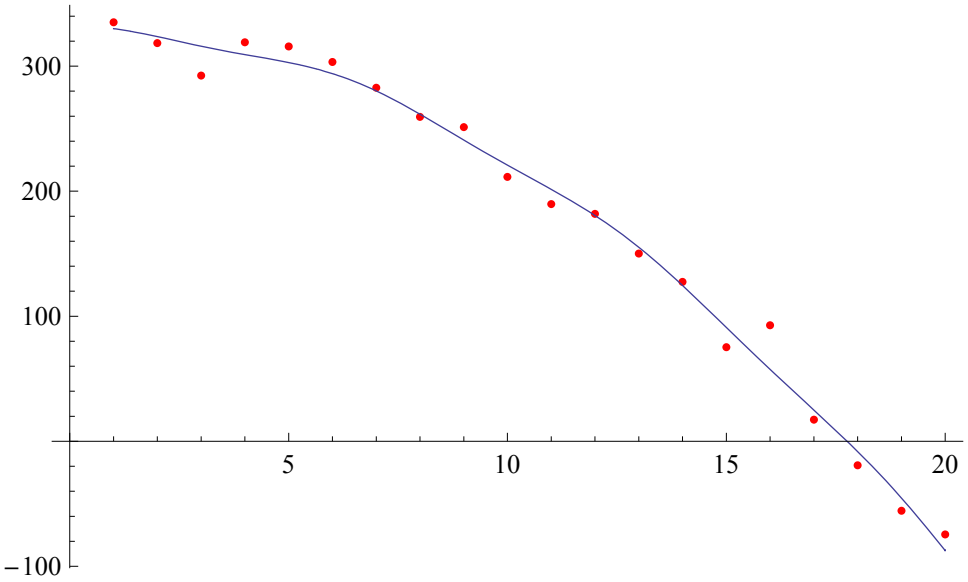
```
In[67]:= 曲線with三角関数 =
          Fit[生成データ, {1, x, x^2, Sin[x], Cos[x]}, x]

Out[67]= 328.905 - 0.0661435 x - 1.04225 x^2 +
          3.41616 Cos[x] + 0.526301 Sin[x]
```

前項までと同じように，組込み関数Showを使うことで重ね合わせて表示させることが出来ます．もちろん，Tableを使ってデータ列をサンプリングすることで，点グラフ

```
In[68]:= Show[グラフ1, Plot[曲線with三角関数, {x, 1, 20}]]
```

Out[68]=



一方，推定すべきパラメータが線形でなく複雑な場合には，組込み関数Fitを使うことが出来ません．このようなときは，「次のモデルのパラメータを指定したデータにフィットするように求めなさい」という意味を持つ組込み関数「FindFit」を使います．

下記では，生成データをFindFitにより，「$a\,x^2 + b\sin(c\,x) + d$」なるモデル（a, b, c, dは未知のパラメータ）への当てはめを行わせています．*Mathematica*は結果をSolveなどと同じく置換規則で返します．

```
In[69]:= FindFit[生成データ, a x^2 + b Sin[c x] + d, {a, b, c, d},
          x]
```

```
Out[69]= {a → -1.04175, b → -1.17478, c → 1.45104, d → 328.294}
```

FindFit[データのリスト，モデル，{a,b,...}，x}]

- モデル：パラメータを含むモデルの式
- {a,b,...}：モデルに含まれるパラメータのリストを指定
- x：目的変数を指定

フィッティング：任意のモデルのパラメータを推定

実際に，モデルの各パラメータに求まった値を代入し，フィッティングによる得られた関数を「曲線by非線形」というシンボルに割り当てましょう．ここでは直前に

```
In[70]:= 曲線by非線形 = a x^2 + b Sin[c x] + d /. %
```

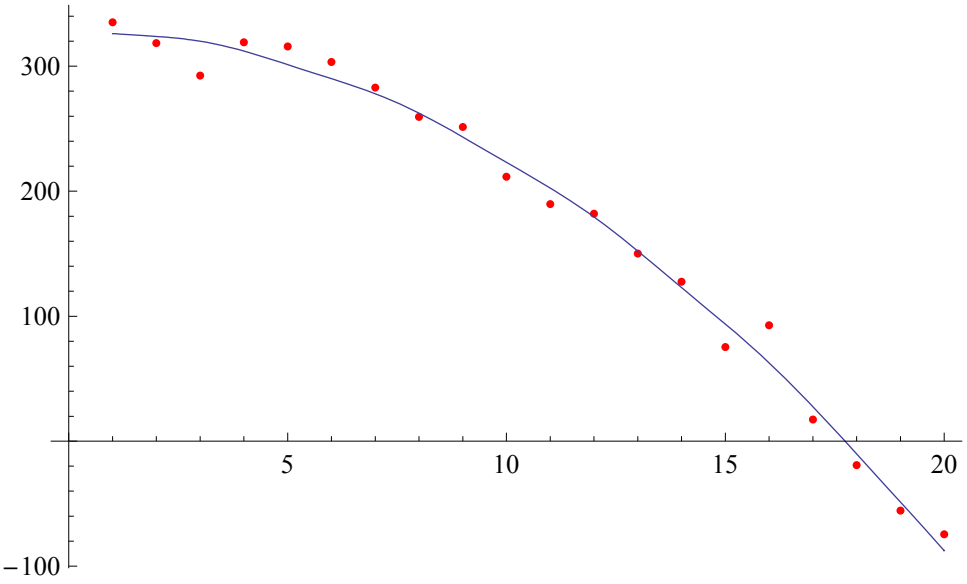
```
Out[70]= 328.294 - 1.04175 x^2 - 1.17478 Sin[1.45104 x]
```

前項までと同じように，組込み関数Showを使うことで重ね合わせて表示させてみます．この例では，線形でも，多項式でも，非線形でも結果は余り変わりません

が，データによってはモデルにより大きく変化します．

In[71]:=
```
Show[グラフ1, Plot[曲線by非線形, {x, 1, 20}]]
```

Out[71]=



【なぜ置換規則で報告するのか】*Mathematica*は，指示された組込み関数により，直接求まった値（数値の場合もあれば数式の場合も）を報告してくることもあれば，今回のFindFitのように置換規則で報告してくることもあります．統一されていた方が分かり易いかもしれませんが，その指示の結果を他の操作に使う可能性の高い命令（パラメータや根を求めるなど）では置換規則の方が，そのような操作が一般に少ない場

数や文字列を*Mathematica*からファイルに出力しよう

■ フィッティングの式からデータ列を作ろう

前項までにも出てきましたが，回帰直線などのフィッティングの結果から，回帰直線上などのデータ列を生成するには，組込み関数のTableを使えば良いということはわ

ここでは，次のようなFindFitによる非線形フィッティングを行って得られた関数に

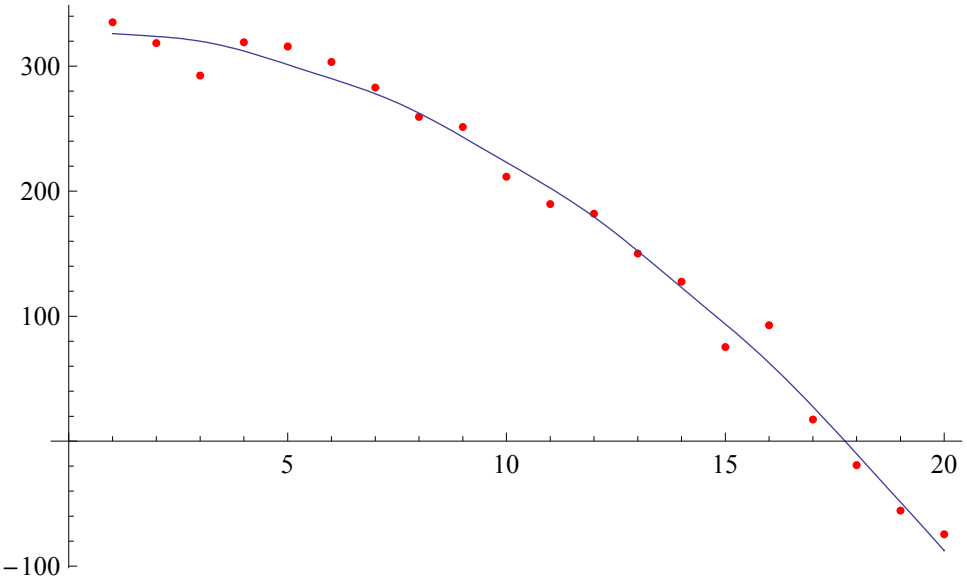
In[72]:=
```
曲線by非線形 = a x^2 + b Sin[c x] + d /.
  FindFit[生成データ, a x^2 + b Sin[c x] + d, {a, b, c, d}, x]
```

Out[72]=
$$328.294 - 1.04175\, x^2 - 1.17478\, \text{Sin}[1.45104\, x]$$

一応，フィッティングで得られた関数をグラフにすると次のようになります．

In[73]:=
```
Show[グラフ1, Plot[曲線by非線形, {x, 1, 20}]]
```

Out[73]=



フィッティングで得られた関数から曲線上のデータ列を得るためには，組込み関数のTableを使います．生成したいデータの形「{x,　曲線by非線形}」を最初に，目的変

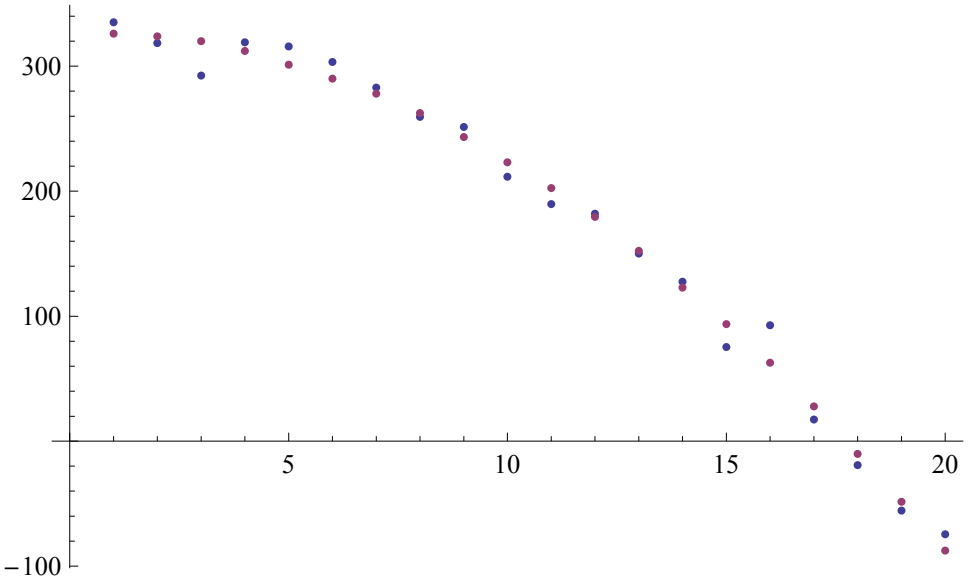
In[74]:=
```
曲線by非線形のデータ = Table[{x, 曲線by非線形}, {x, 1, 20}]
```

Out[74]=
```
{{1, 326.086}, {2, 323.848}, {3, 320.018}, {4, 312.168},
 {5, 301.28}, {6, 290.018}, {7, 278.034}, {8, 262.583},
 {9, 243.357}, {10, 223.025}, {11, 202.537},
 {12, 179.447}, {13, 152.223}, {14, 122.944},
 {15, 93.6383}, {16, 62.7124}, {17, 27.7562},
 {18, -10.2113}, {19, -48.5374}, {20, -87.6069}}
```

このようにして作ったデータを元のデータと確認したい場合は，点グラフを描かせる

In[75]:=
```
ListPlot[{生成データ, 曲線by非線形のデータ}]
```

Out[75]=

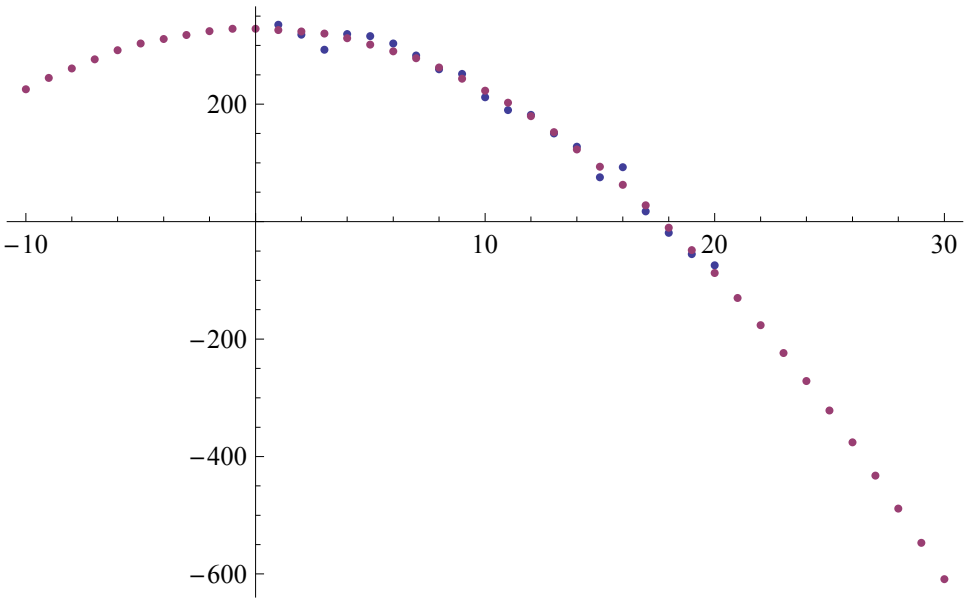


既にフィッティングで関数が求まっているので，それを使った外挿も可能です．単純にTableで指定する目的変数の範囲を広げることで，フィッティングされた関数によ

In[76]:=
```
曲線by非線形のデータ外挿 =
  Table[{x, 曲線by非線形}, {x, -10, 30}];
```

In[77]:=
```
ListPlot[{生成データ, 曲線by非線形のデータ外挿}]
```

Out[77]=

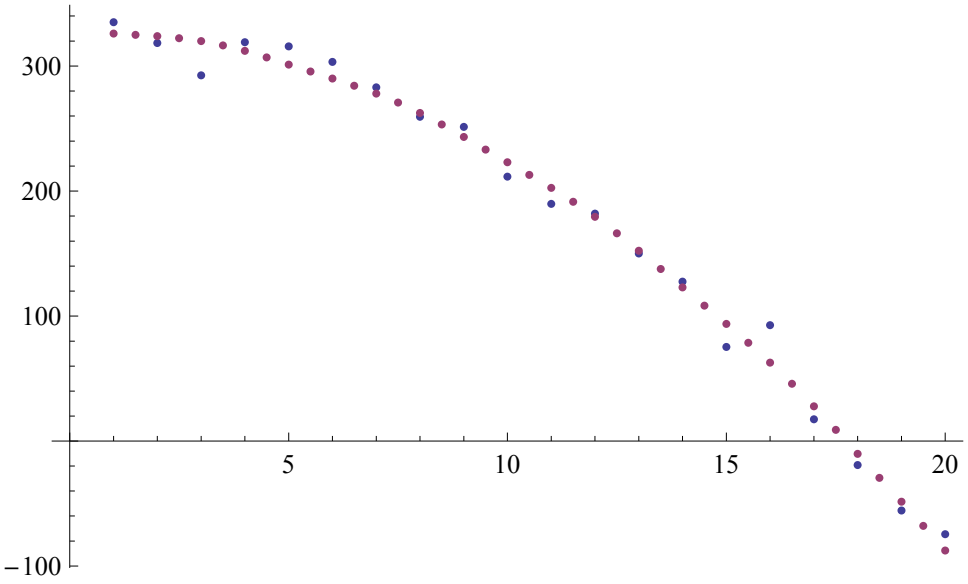


一方，内挿を行う場合には，目的変数の範囲を指定しているリストの最後に刻幅を加えます．これまでは刻幅が1だったので何も指定していませんでしたが，範囲指定の最後には刻幅を指定できます．以下の例では，刻幅を0.5にすることで内挿を行ってい

In[78]:=
```
曲線by非線形のデータ内挿 =
  Table[{x, 曲線by非線形}, {x, 1, 20, 0.5}];
```

In[79]:=
```
ListPlot[{生成データ, 曲線by非線形のデータ内挿}]
```

Out[79]=



【この項の意味】*Mathematica*はグラフを描くのに，データ列を必要としません．従って，本項で扱ったデータ列の作成は，グラフを描かせるためではなく，内挿や外挿を行うためであったり，その結果を次項で扱うように表計算ソフトに出力するためのも

■ **フィッティングの結果を表計算ソフトに出力しよう**

この章の冒頭で*Mathematica*に取り込んだ表計算ソフトのデータに対して，次のように非線形フィッティングを行った場合を考えましょう．ここでは「」というモデルでフィッティングをFindFitで行い，結果を「最初のシートの関数」というシンボルに割

In[80]:=
```
最初のシートの関数 =
  a x^2 + b Sin[c x] + d /.
   FindFit[Drop[最初のシート, 1], a x^2 + b Sin[c x] + d,
    {a, b, c, d}, x]
```

Out[80]=
$$2.08469 + 1.00287\,x^2 - 0.422091\,\text{Sin}[0.961878\,x]$$

前項の方法に基づいて，次のように内挿を行います．結果はかなり長くなるので，ここではセミコロンを付けて出力を抑制しています．結果は「内挿」というシンボルに

In[81]:=
```
内挿 = Table[{x, 最初のシートの関数}, {x, 1, 20, 0.25}];
```

このままでは数値データしかなく不便なので，最初のシートの最初の要素（これは次のようにFirstで取り出せます）をデータ列の最初に加えておきましょう．こうする

```
In[82]:= First[最初のシート]
Out[82]= {実施日, 測定値}
```

データ列が納められているシンボル「内挿」はリストになっているので，その項目名を最初に追加するには，「次のリストの最初に指定した要素を追加しなさい」という

```
In[83]:= 内挿withラベル = Prepend[内挿, First[最初のシート]];
```

Prepend[リスト, 追加したい要素]

この最初に追加する

最初に入れたい要素

要素の追加：リストの最初に要素を追加する

このようにして内挿したデータの形式を表計算ソフトから読み込んだデータと同じ形式にしました．表計算ソフトへのデータの出力は「次のデータをファイルに適切な形式で保存しなさい」という意味の命令「Export」を使います．Exportは指定されたファイル名の拡張子によって，適切な形式でファイルにデータを保存しますので，今

```
In[84]:= Export["interpolate_vertical.xls", 内挿withラベル]
Out[84]= interpolate_vertical.xls
```

Export[出力先のファイル名, データ]

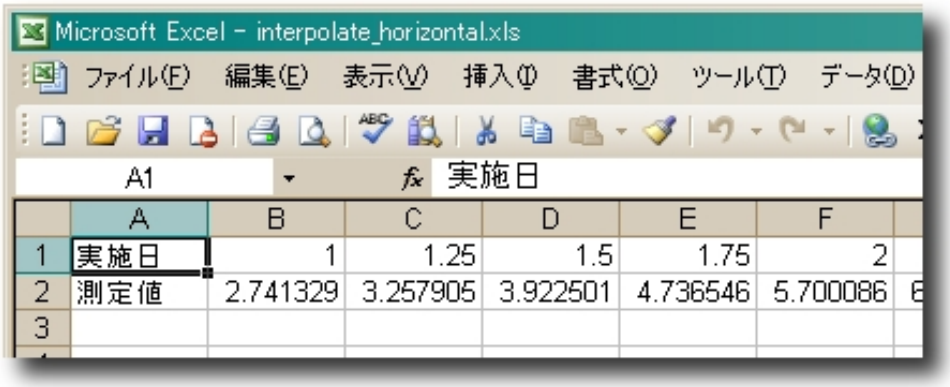
適切な拡張子を持つファイル名にする

ファイルに保存したいデータを指定

ファイルへの保存：表計算ソフトに*Mathematica*のデータを出力

表計算ソフト上で横に並ぶデータとして保存したい場合は，事前に縦と横を入れ換える命令Transposeで縦と横を入れ換えておきます．このようにすることで，下記の図

```
In[85]:= Export["interpolate_horizontal.xls",
           Transpose[内挿withラベル]]
Out[85]= interpolate_horizontal.xls
```



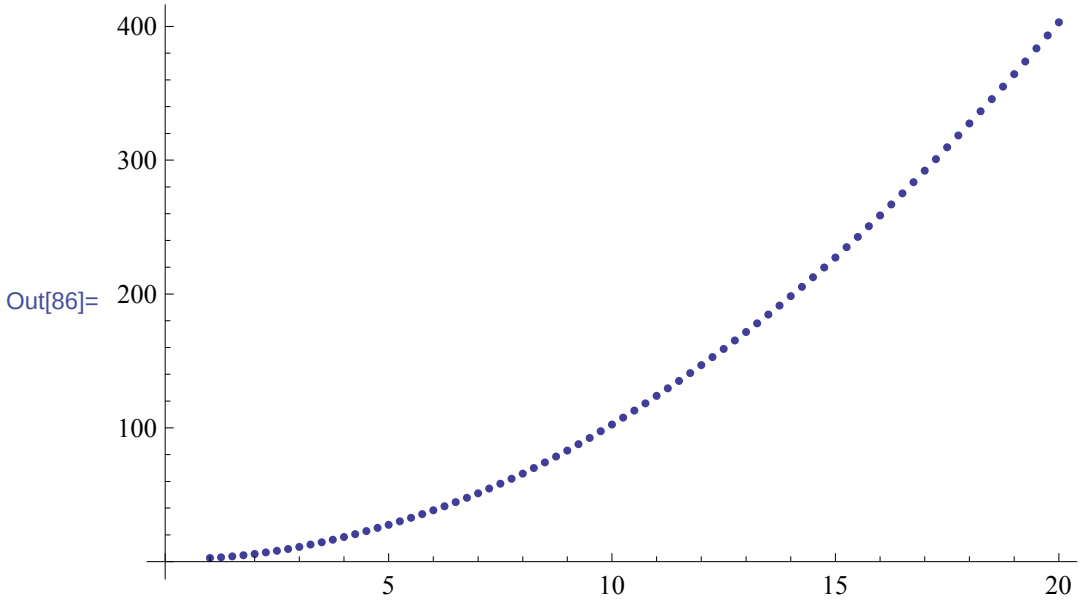
表計算ソフト：内挿結果を出力したExcelファイルの内容

- **グラフを画像ファイルやPDFに出力しよう**

*Mathematica*では数値データだけでなく画像なども様々なファイル形式で出力することが出来ます．既に学んだように，これらは画像等を選択してファイルメニューから保存出来ますが，表計算ソフトへの出力と同じく組込み関数のExportを使っても可能です．一度に複数のグラフを出力したい場合などは，メニューを使うよりもExportで出力する方が楽です．

ここでは実際に次のようなグラフをファイルに保存することを考えましょう．ファイルに保存すれば，様々なアプリケーションソフトからも利用出来るので便利です．もちろん，コピー&ペーストでも他のアプリケーションで利用できますが，画像ファイル単体にしておく方が記録上は便利と思います．

```
In[86]:= 内挿のグラフ = ListPlot[内挿]
```



画像ファイルへの保存の仕方も表計算ソフトへの保存の方法と同じです．唯一異なる点は，出力先のファイル名の拡張子が画像ファイルの拡張子となることです．次の例では「.jpg」と拡張子を指定しているので，JPEG形式の画像ファイルとして保存され

ます．この他，「.png」でPNG形式，「.bmp」でBMP形式など，多くの画像形式で保存することが出来ます．

In[87]:=
```
Export["interpolate.jpg", 内挿のグラフ]
```

Out[87]=
```
interpolate.jpg
```

また，拡張子を「.pdf」とすることで，PDF形式でも保存できます．

In[88]:=
```
Export["interpolate.pdf", 内挿のグラフ]
```

Out[88]=
```
interpolate.pdf
```

【ファイル形式】*Mathematica*は実に様々なファイル形式に対応しています．この項で利用したExportという命令は，表計算ソフトや画像だけでなく様々なデータ形式のファイルとして保存することが出来ます．扱える形式は「$ExportFormats」という命令で確認できますが，100種類を超えています．