

Doppler Tomography using Total Variation Minimization (DTTVM): User Guide

Makoto Uemura (Hiroshima University)

6 Oct. 2025

This document explains how to use DTTVM (Doppler tomography with total variation minimization). The code is written in C, and developed on Linux. If you want to try it right now, read “Tutorials” below. After the tutorials, details about the compile of the source code, model parameters, the format of the input file, output products, and how to see the outputs are shown in this document. For details about the model, see Uemura, et al. (2015).

Ver. 1.11 (6 Oct 2025)

The current version, including the Python scripts for drawing the secondary Roche lobe, stream, etc, in the Doppler map. No change in `dtvm`.

Ver. 1.1 (14 Apr 2015)

The second version, including the function for cross-validation (`dtvm_cv`) and the tool for drawing the secondary Roche lobe, stream, etc, in the Doppler map. No change in `dtvm`.

Ver. 1.0 (12 Jul 2013)

The first version of `dtvm`, manual document, demo data, and sample R script for drawing the results.

1 Tutorials

When you read this document, you have already downloaded the file, `dtvm.tar` and unpacked it. You can try to run `dtvm` by following commands, in the directory you unpacked it,

```
$ cd dtvm
$ gcc -o dtvm dtvm.c -lm
$ ./dtvm demo.tumen.dat 64 100 0.01
```

After convergence, you will find two output files, “`dopmap_demo.tumen.dat`” and “`trail_demo.tumen.dat`”. You can see the results using Python.

For Python users, Doppler maps can be drawn as follows:

```
import dopmap
import matplotlib.pyplot as plt
fig, ax = dopmap.dopmap("dopmap_demo_tumen.dat")
dopmap.draw_2ndlobe(ax, q=0.455, incl=52, porb=0.117, m1=0.785, color="white")
dopmap.draw_stream(ax, q=0.455, incl=52, porb=0.117, m1=0.785, color="white")
# plt.savefig("dopmap_demo_tumen.pdf")
plt.draw()
plt.show()
```

Trailed spectra can be drawn as follows:

```
fig, ax, img = dopmap.trail("trail_demo_tumen.dat", value='model', nphase=20,
nvel=50)
# plt.savefig("trail_demo_tumen.pdf")
plt.show()
```

value can be "data", "model", and "residual".

2 Compile

To compile `dtvm` and `dtvm.cv` is simple. Just do,

```
$ gcc -o dtvm dtvm.c -lm
$ gcc -o dtvm.cv dtvm.cv.c mt19937.c gasdev.c -lm
```

3 Parameters

You can see required parameters for `dtvm` when you just type `dtvm` at the command line.

```
$ dtvm
Usage: dtvm data_file nbin resol lambda (Vgamma thresh)
For help: dtvm -h
```

`data_file` is a file containing spectroscopic data. It is a set of three-column data: the radial velocity (measured from the center of mass), normalized flux, and orbital phase. The format is described in the next section.

`nbin` is the number of bins in a side of the Doppler map. The total number of bins in the Doppler map is $N = \text{nbin} \times \text{nbin}$.

`resol` is a velocity resolution of the input spectra, in km/s. It is not equal to the velocity per pixel, but the minimum velocity interval that can be resolved with the instruments. It is assumed that the signal at a radial velocity is smeared out with a Gaussian having a standard deviation of "resol" in the model.

`lambda` is a hyperparameter of the model. It is a weight for the total variation (TV) term. A smaller “lambda” generates a more noisy map, while a larger “lambda” generates a featureless map. Typically, $10^{-5} - 10$.

`Vgamma` is a gamma velocity in km/s (optional). It is subtracted from the radial velocity listed in the input file. Default value is 0.0.

`thresh` is a threshold to terminate the TwIST iteration (optional). The iteration is terminated when the fraction of the current to last objectives is less than this value. Default value is 10^{-4} . Typically, $10^{-3} - 10^{-5}$.

Examples:

In the last section, you might have tried:

```
$ ./dtvm demo_tumen.dat 64 100 0.01
```

The most important parameter may be “lambda”. Check the resultant maps by changing it:

```
$ ./dtvm demo_tumen.dat 64 100 0.1
```

```
$ ./dtvm demo_tumen.dat 64 100 0.001
```

Increasing the bin number of the Doppler map:

```
$ ./dtvm demo_tumen.dat 128 100 0.01
```

Check the dependence of the value of the velocity resolution:

```
$ ./dtvm demo_tumen.dat 64 200 0.01
```

Check the dependence of the value of the gamma velocity:

```
$ ./dtvm demo_tumen.dat 64 100 0.01 100
```

The default value of “thresh” may be too large to obtain the result for publication. In some cases, you can try smaller values:

```
$ ./dtvm demo_tumen.dat 64 100 0.01 0 1e-5
```

4 Input file

The data file consists of three columns: radial velocity in km/s, normalized flux, and orbital phase. The data is supposed to contain only one emission-line component. Multiple and overlapped profiles are not supported. The continuum level should be zero in the normalized flux.

The radial velocity is supposed to be measured from the center of mass, while the γ -velocity can be corrected with the optional parameter, `Vgamma` of `dtvm`.

In most cases, 2D images are reduced to be 1D spectra using some softwares, like IRAF. Then, the 1D spectra are normalized by dividing by the continuum level. The input data can be obtained by subtracting 1.0 from the normalized spectra. Add the orbital phase to each spectra, and finally combine all spectra

to make the input file.

For example, the data file is as follows:

```
-1973.533 0.2882 0.0500884
-1923.158 -0.007948 0.0500884
-1872.737 0.221362 0.0500884
-1822.317 0.38597 0.0500884
...
1856.829 0.089344 0.0500884
1907.204 0.379797 0.0500884
1957.625 0.1251 0.0500884
-1973.761 0.145837 0.1189197
-1923.341 0.643033 0.1189197
-1872.966 0.623131 0.1189197
...
```

5 Outputs

After convergence, you will find two output files, “dopmap_xxx.dat” and “trail_xxx.dat” in the case that the input file is named “xxx.dat”. In both output files, the parameters used for calculation are listed at the first line. Those two output files can be viewed with R, as described in the next section.

The file “dopmap_xxx.dat” is for the resultant Doppler map. It contains three columns: V_x , V_y , and intensity.

The file “trail_xxx.dat” is for the data of spectra. It contains four columns: the radial velocity, observed spectrum, model spectrum, and residuals between the observation and model. The order of the data is the same as the input file.

The process of iteration is shown in the terminal (as stderr) during calculation. For example:

```
Iteration= 2, objective=8.57605e+02, criterion=5.815e+04
Iteration= 3, objective=3.04327e+02, criterion=6.451e+04
Iteration= 4, objective=2.03092e+02, criterion=3.327e+04
...
Iteration= 211, objective=6.22012e+01, criterion=1.023e+00
Iteration= 212, objective=6.22006e+01, criterion=9.910e-01
Iteration= 213, objective=6.22000e+01, criterion=9.554e-01
```

The first column shows the number of iteration. The second is the value of the objective function. The problem which is solved by this code is to find the minimum of the objective function. The third is the ratio of the current and past values. The iteration is stopped when it becomes less than 1.0. It is defined as $(\text{obj}_i - \text{obj}_{i-1})/\text{obj}_{i-1}$. In some cases, you may see the message like:

Max_svd revised : xxx

This is just a report to revise an internal parameter in the code.

6 Graphhics with Python

The output files, “dopmap_xxx.dat” and “trail_xxx.dat” can be visualized with Python. The Python code, “dopmap.py” is also included in the file “dtvm.tar”. It has two main function, “dopmap()” and “trail()”.

They are not high-quality codes. You can use other graphic tools or software if you want to see the results with more options. The Python codes only provide minimal tools to see the results.

The “dopmap” is a function to draw Doppler maps, and defined as:

```
dopmap(file,...)
```

The “file” is a dopmap file generated by dtvm.

You can use some useful curves on the Doppler map:

`draw_2ndlobe(ax, q, incl, porb, m1=0.6, vfs=1e5,...)`: Drawing the Roche lobe of the secondary.

`draw_stream(ax, q, incl, porb, m1=0.6, n=250, vfs=1e5,...)`: Drawing the stream path from $L1$.

`draw_kepler(ax, q, incl, porb, m1=0.6, n=250, vfs=1e5,...)`: Drawing the Kepler velocity along stream.

`draw_circle(ax, r, x0, y0,...)`: Drawing the circle with radius r centered at $x0, y0$.

`draw_center(ax,...)`: Plot “+” at the center of mass.

`draw_m2(ax, q, incl, porb, m1=0.6, vfs=1e5,...)`: Plot “x” at the position of the secondary (M_2).

`draw_m1(ax, q, incl, porb, m1=0.6, vfs=1e5,...)`: Plot “x” at the position of the primary (M_1).

Several parameters in those drawing functions are common binary-parameters or drawing configurations:

`q`: Mass ratio (M_2/M_1).

`incl`: Inclination angle. Pole-on= 0.0. Edge-on= 90.0.

`porb`: Orbital period in days

`m1`: Mass of the primary in solar-mass. Default is 0.6.

`vfs`: Velocity scale factor. cm/s= 1.0. Default is km/s= 10^5 .

You can draw the curves and symbols with any colors, for example:

```
draw_2ndlobe(ax, 0.1, 75.0, 0.55, color='white')
```

The “trail()” is a function to see spectra, and defined as:

```
trail(file, value='data', nphase=20, nvel=50, cmap='gray_r', wrap_phase=True,
flip_phase=True, show=False)
```

Parameters

```
file : str
Input file path. Columns: v, sp0, sp1, phase
value : str
'data', 'model', or 'residual'
nphase : int
Number of bins along phase
nvel : int
Number of bins along velocity
cmap : str
Matplotlib colormap
wrap_phase : bool
If True, phase values  $\geq 1$  are wrapped ( $ph - 1$ )
flip_phase : bool
If True, phase 0 is at the bottom of the plot
show : bool
Whether to call plt.show() at the end
```

7 Cross-validation

dtvm has a tuning parameter, λ , which controls the sparsity in the differential domain of the Doppler map. The optional function, `dtvm_cv` offers a method to estimate the best λ for the data using cross-validation (CV). For details, see Uemura, et al. (2015).

You can see required parameters for `dtvm_cv` when you just type `dtvm_cv` at the command line.

```
$ dtvm_cv
Usage: dtvm_cv data_file nbin resol lambda_min lambda_max nlambda
(Vgamma thresh)
For help: dtvm_cv -h
```

The results are output in stdout. In the case of the demo data, `demo_tumen.dat` (See Section 1 Tutorials), it works like:

```
$ ./dtvm_cv demo_tumen.dat 64 100 0.001 10 20 > cv_tumen.dat
```

`data_file`, `nbin`, `resol`, `Vgamma`, `thresh` are the same as those for `dtvm`. `lambda_min`, `lambda_max` are the minimum and maximum λ between which the best λ is searched.

nlambda is the number of lambda for CV.

dtvm_cv performs dtvm 10 times for each lambda. Hence, the total number of the run of dtvm is $10 \times \text{nlambda}$ in dtvm_cv. Users should estimate the time for calculation before running dtvm_cv. If it teaks ~ 1 min for one run of dtvm, then it takes ~ 100 min for a run of dtvm_cv with nlambda= 10.

The status of calculation is shown in stderr, like:

```
Lambda ID 01/20 Sub-Set ID 01/10 (Nt=1072, Nv=113) err=1.267746e-01
Lambda ID 01/20 Sub-Set ID 02/10 (Nt=1088, Nv=97) err=1.622698e-01
Lambda ID 01/20 Sub-Set ID 03/10 (Nt=1063, Nv=122) err=1.311803e-01
...
```

In the above case, 20 lambdas are calculated, and 10 calculations are performed for each lambda. The value, err is the RMS values, which should take the minimum between 20 lambdas.

The output file (cv_tumen.dat, in the case of the above example) contains three columns, that is, lambda, RMS, and its standard error.