Chapter 2

The Carlifornia Planet Survey Doppler Code

This chapter contains a brief documentation describing the algorithm and structure of the California Planet Survey (CPS) Doppler code, which extracts RVs from iodine-calibrated stellar spectra. As of March 2016, no documentation in published or unpublished form existed for this widely used code, although Butler et al. (1996a) describes the basics for the technique of iodine-calibrated precise RV, and some CPS publications contain description for certain elements of the code (e.g., Johnson et al. 2006; Howard et al. 2009, 2011; Johnson et al. 2011a).

The earliest date mark in the code is 1991, which is roughly when Paul Butler and Geoffrey Marcy started drafting the code. The code was heavily modified by John Johnson around 2002-2008. Later on, two versions of the code are maintained by two separate groups: the California Planet Survey (LCPS) team (John Johnson) and the Lick-Carnegie Planet Survey team (Paul Butler). Post 2014, the CPS version of the code was maintained mostly by Howard Isaacson at UC Berkeley. This code is widely used by many iodine-calibrated precise RV instruments, such as Lick/Hamilton, HET/HRS (this thesis; Chapter 3), AAT/UCLES, Magellan/PFS, and the Automated Planet Finder (APF) at Lick (both LCPS and CPS have their own version of this code for APF). Our copy of the code was kindly provided by John Johnson in 2009, and the copy was checked to still be consistent with the CPS version in 2013.

2.1 Basic Formulae, Algorithm, and Components

First, we describe the basic mathematics and algorithm behind RV extraction from iodine-calibrated stellar spectra using the CPS code. The overall algorithm is to forward model the stellar spectra using synthetic or empirically derived reference spectra, finding best-fit model parameters including the Doppler shift, z.

The reference spectra include[1]: a model spectrum for the iodine absorption lines, $F_{\mathrm{I}_2}(\lambda)$ and a model spectrum for the star, $F_\star(\lambda)$. The goal is to use the model the

[1]It can also include a model spectrum for a faint secondary star, telluric absorption lines (see Chapter 4 Section 4.1), and so on.

Table 2.1. Parameters for Forward Modeling Keck/HIRES RV Spectra

Parameter	Unit and Meaning
z	no unit, the stellar red shift
w_0	Å, wavelength of the first pixel of a spectral chunk
w_d	Å/pixel, wavelength dispersion scale for a spectral chunk
$A_n,\ n = 1, ..., 12$	no unit, amplitudes of side gaussians for IP[a]

[a]See Section 2.1.2 for more information.

observed, extracted, and normalized 1-D spectrum, $F_{\text{obs}}(x)$, at any given pixel position (and spectral order), x, using these reference spectra and model parameters. The broadening effect of the spectrograph is described by the spectral response function, or the spectral point spread function, or the instrumental profile (IP), which we will refer to as the IP throughout this thesis and is denoted as $\mathcal{P}(x)$. Hence,

$$F_{\text{obs}}(x) = \left[F_{\text{I}_2}(\lambda(x)) \times F'_\star(\lambda(x))\right] * \mathcal{P}(x), \tag{2.1}$$

where $\lambda(x)$ is the wavelength solution for the 1-D spectrum, and $F'_\star$ is the red-shifted stellar spectrum defined by $F'_\star(\lambda) = F_\star(\lambda \cdot (1+z))$. The Doppler shift z contains two components: the stellar RV $v_\star$ and the barycentric (BC) velocity of the Earth v_{BC}. The BC component is corrected by $v_\star = v_{\text{measured}} - v_{\text{BC}} + z \cdot v_{\text{BC}}$.

The stellar reference spectrum is empirically derived from iodine-free stellar observations taken on an epoch, say, T_0. As a result, all measured RVs for the star using a stellar template from T_0 represent relative stellar velocities between epoch T_0 and epoch T_{obs} (i.e., $v_{\star,T_{\text{obs}}} - v_{\star,T_0}$), instead of the absolute~~ly~~ RV~~s~~ of the star. The following subsection describes the origins of the stellar (and also the iodine) reference spectra.

In practice, for Keck/HIRES, for example, each 1-D spectrum taken at an epoch is divided into ~700 spectral chunks, each with 80 pixels and about 2Å in wavelength. One model is created and fitted for each spectral chunk, with the model parameters listed in Table 2.1. Model parameter optimization is done through least-χ^2 fitting using the Levenberg-Marquardt (LM) algorithm. Errors on the extracted 1-D spectrum are assumed to be Poisson noise plus a 2% additional representing potential errors in the raw reduction. Initial guesses of the parameters come from the solution for nearest B star + iodine observation, with the exception of the initial guess for z, which is set to be v_{BC} because that is usually on the order of km/s and dominates the Doppler shift signal.

To sample the reference spectra into the observed pixel grid, the code first re-samples (using spline) each of them onto a grid finer than the observation pixel grid by a factor of four (i.e., using a wavelength dispersion of $w_d/4$). The wavelengths of this fine grid is provided by the proposed wavelength solution parameters w_0 and w_d. Next, it shifts the stellar reference spectrum according to the proposed Doppler shift parameter z. Then

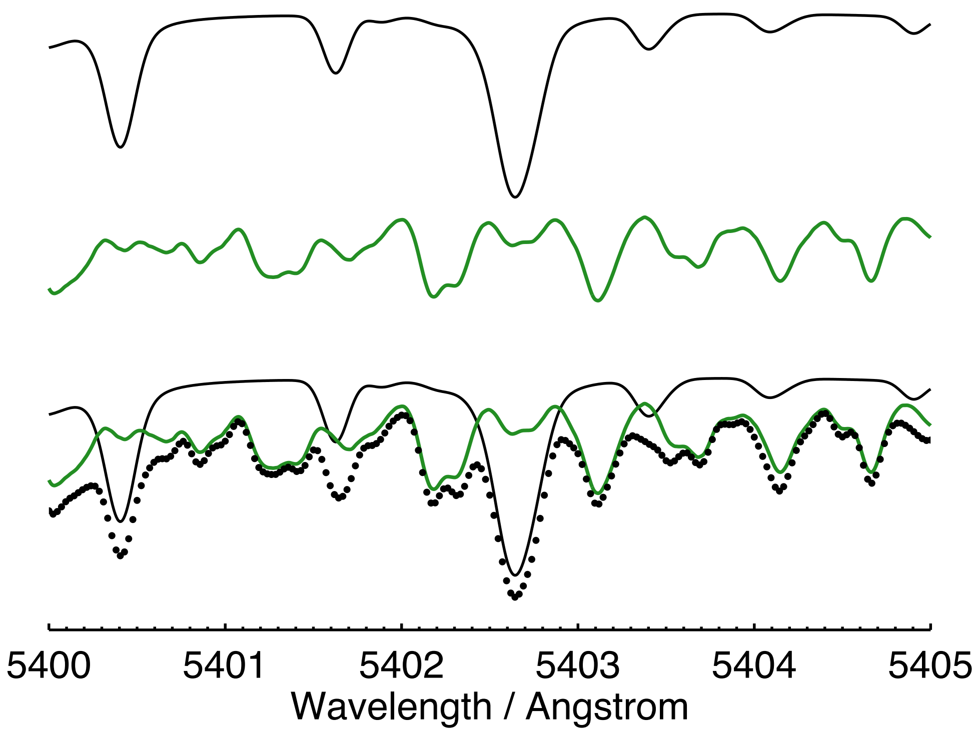

Figure 2.1 An illustration for the forward modeling process for iodine-calibrated stellar spectrum. The top black line represents the stellar reference spectrum (DSST), and the middle green line represents the iodine reference spectrum. Both reference spectra are convolved with an IP only for illustration purposes in order to have a clear match with the observed spectrum, plotted in black dots on the bottom. In practice, the reference spectra are multiplied *first* and then convolved with IP.

it multiplies the shifted stellar reference spectrum with the iodine reference spectrum, and then convolves the product spectrum by the IP. Finally, it re-bins the finely sampled model spectrum onto the observed pixel grid, which yields the final normalized model spectrum $F_{\rm model,norm}(x)$.

In reality, the observed spectrum used in the fitting is not normalized (i.e. no blaze or continuum removal). To account for blaze and stellar continuum, the codes divides the observed spectrum by the normalized model spectrum, i.e., $F_{\rm obs}(x)/F_{\rm model,norm}(x)$, and then it fits a straight line $S(x)$ through the divided spectrum (for a small 2Å chunk, this linear approximation seems sufficient). It then computes a new model spectrum by adding this model "continuum" on top, i.e., $F_{\rm model,final}(x) = F_{\rm model,norm}(x) \times S(x)$. This way, the continuum component in the observed spectral chunk is modeled by a linear function but imposes no explicit parameters for the model.

2.1.1 The Reference Spectra

Ideally, the reference spectra are the "ground truth" spectra, i.e. the intrinsic spectra of the sources (e.g., the iodine cell, or the star) without Doppler shift or being broadened by the spectrometer. In reality, there is no way of knowing such "ground truth", so the reference spectra are empirically derived from observations.

The iodine reference spectrum, often referred to as the iodine atlas, originates from a Fourier Transform Spectrometer scan of the iodine cell illuminated by a continuum source. It is often of very high signal-to-noise ratio (SNR) with high resolution (normally $\sim 500,000$ or larger). Therefore, it is generally regarded as basically the "ground truth" for the cell, especially for the purpose of forward modeling lower-resolution ($\sim 60,000$) spectra. However, there can be problems with the iodine atlas, for various rea[illegible]. See Chapter 3 Section 3.2 for more on this topic. The current FTS iodine atlas being used for Keck/HIRES RV work is from a scan in 1993, using the Babar FTS at NSO/KPNO, and so is the atlas for HET/HRS. See Section 3.2 for more on iodine reference spectra.

The stellar reference spectrum for any star, or internally to CPS referred to as the Deconvolved Stellar Spectral Template (DSST), is empirically derived from observed spectra of the target star. For most of the CPS targets (bright stars), a few (4-5) observations of the star with a narrower slit ($R \geqslant 80,000$) are taken without the iodine cell in the light path. Then they are stacked together to boost the SNR (> 500), and then deconvolved with proper IPs derived from bracketing B star + iodine observations. The wavelength solution for the DSST also comes from the bracketing B star + iodine observations. See Section 4.2 for more information and problems related to DSSTs. For faint stars where obtaining stellar template is expensive or unfeasible, Johnson et al. (2006) developed a technique where they "morph" a synthetic stellar spectrum or an existing DSST of another star with similar stellar properties to fit the stellar iodine observation, and then they use this new morphed DSST for RV extraction.

2.1.2 The Functional Forms of the Instrumental Profile

The IP $\mathcal{P}(x)$ can take many functional forms, and for Keck/HIRES, an IP of sum of gaussians works exceptionally well ($\chi^2 \sim 1$ for pure iodine absorption line fit). The mathematical form for it is:

$$\mathcal{P}_{\text{gaus}}(x) = \sum A_n \exp\left[\left(\frac{x - \mu_n}{\sigma_n}\right)^2\right]. \tag{2.2}$$

A_n stands for the amplitude for each gaussian component. A_n's are floated parameters for the fitter to optimize while μ_n and σ_n (i.e., positions and widths of the gaussians) have empirically-optimized fixed values, depending on the instrument setting of Keck/HIRES (e.g., slit width). For Keck/HIRES precise-RV mode (B5 decker, ∼60,000 resolution, with iodine cell in light path), the IP contains 12 free parameters, $A_1, A_2, ..., A_{12}$, while A_0 is fixed to 1 (the big central gaussian) and $\mu_n, n = 0, ..., 12$ and $\sigma_n, n = 0, ..., 12$ also have fixed values.

Another frequently used IP is the Gauss-Hermite (GH) function, which is composed of gaussians multiplied by Hermite polynomials H_n:

$$\mathcal{P}_{\mathrm{GH}}(x) = \sum A_n u_n(x) = \sum A_n \left(\frac{2}{\pi w^2}\right)^{1/4} \frac{1}{\sqrt{n! 2^2}} H_n\left(\frac{\sqrt{2}x}{w}\right) \exp\left[-\left(\frac{x}{w}\right)^2\right]. \quad (2.3)$$

Mathematically, any sum of gaussians can be decomposed into orthogonal GH terms[2], and therefore, in principle, the GH IP should present a generic and flexible option for IP choices. However, in reality, the least-χ^2 solver is extremely sensitive to the choices of initial guesses, even for orthogonal bases. As a result, GH IP normally does not outperform sum of gaussians (e.g., see work by Vanderburg et al. 2013). The GH IP is what we use for extracting RVs from HET/HRS data. See Chapter 3 Section 3.3 for more.

2.2 Code Structure and Work Flow

This section documents the structure of the CPS Doppler code, with the main goal to help any reader who wishes to adopt this code for their own work. Figure 2.2 illustrates the code's calling sequence.

The most top-level routine for running Doppler reduction is the IDL procedure `dop_driver.pro`, which takes in the name of the star and then automatically locate input files such as the extracted 1-D spectra, the proper DSST file, the iodine atlas, and the files storing initial guesses for parameters (which are called the `vdiod` files). The code is also very flexible with inputs and the user can specify almost anything, e.g., a specific DSST file, choice for a specific IP model, explicit initial guesses for parameters, and so on. It drives the Doppler analysis and output a `vd` file for each observed spectrum, which contains best-fit parameters and other information for all the spectral chunks. At the end, `dop_driver.pro` calls `jjvank.pro`, which combines the RVs from all the chunks in all the observations for this star, and evaluates numerical weights for each chunk and each observation. Finally, `jjvank.pro` computes the weighted RV for each observation and its RV uncertainty and outputs the information in `vst` files. The most useful variable is the `cf3` structure in the `vst` file, which contains, for example, the Julian Date (JD) of the observation `cf3.jd`, the BC `cf3.bc`, the weighted RV `cf3.mnvel`, and its uncertainty ~~tt~~ cf3.errvel. The algorithm for `jjvank.pro`, or "vanking", is described in the next section. In the end, `dop_driver.pro` will produce N_{obs} `vd` files and one `vst` file for each star. If a new observation is taken, a new `vd` file is created, and vanking re-evaluates the chunk and observation weights and outputs a new `vst` file.

While `dop_driver.pro` is the routine to call for Doppler analysis, most of its actual codes simply deal with logistical work such as locating the right files. The real driver behind the scene is `crank.pro`, which contains the loop through all observations and "fills out" the `vd` files. In a standard CPS reduction routine, `crank.pro` is called three times. In the case of Keck/HIRES standard RV reduction, for example, the first time `crank.pro` is called, it fits for all 15 free parameters (z, w_0, w_d and 12 IP parameters;

[2]See http://math.stackexchange.com/questions/28719/how-to-decompose-displaced-hermite-gauss-function-into-higher-order-hgs for an illustration, retrieved on March 18 2016.

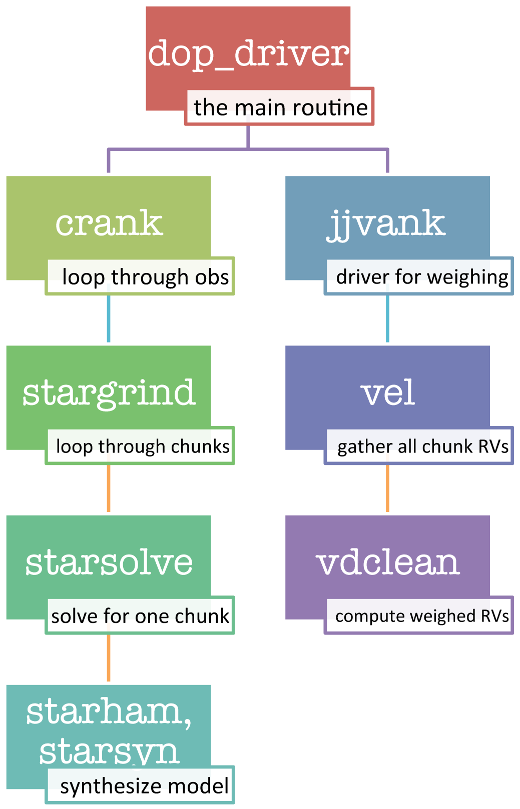

Figure 2.2 Calling sequence and main functionality for core IDL routines in the CPS Doppler code.

see Table 2.1) for each chunk in each observation. The LM fitter takes in the initial guesses for parameters from B star + iodine solutions and tries its best to optimize the 15-parameter model. Due to the complexity of this multi-modal and multi-dimensional problem, very often the best fit for this "first pass" does not yield a good global minimum on the χ^2 surface. Therefore a second pass of `crank.pro` is called, with fixed IP parameters and thus only three free parameters. The IP parameters are not simply fixed to last round's best-fit values, but instead, they were fixed to describe an "averaged" IP over a region on the CCD chip (for example, over a neighbor of nine chunks across three spectral orders for the central chunk). After the second pass, a third pass of `crank.pro` is called with fixed wavelength dispersion w_d and only floating z and w_0, in search for a deeper minimum or lower χ^2.

Within `crank.pro`, the code calls for `stargrind.pro` in a loop of all observations. `stargrind.pro` does the model fitting for a single observation, and it loops through all spectral chunks, calling `starsolve.pro`, which contains the LM fitter, for each chunk. Inside `starsolve.pro`, the spectral model is computed using `starham.pro`, which is really just a wrapper around `starsyn.pro`. The core algorithm for model construction is all in `starsyn.pro`, which takes in the observed 1-D spectral chunk, the DSST, the iodine atlas, and the model parameters, and outputs a model spectrum for this chunk.

2.3 Adjusting Offsets and Computing Weights for Chunk RVs

Now all observations and all chunks have their best-fit model parameters computed by `crank.pro`, and all RVs are barycentric corrected. What's next?

Two extremely important things need to happen at this point before we can have an RV time series in hand, and they are accomplished by `jjvank.pro` and most importantly, `vel.pro` and `vdclean.pro` (see Figure 2.2 for the calling sequence).

First, the chunk RVs need to be adjusted so that they have the same "zero point". Ideally, measured RVs from all chunks in one observation are good estimates for the true RV of this epoch, so the mean of all chunk RVs provides an unbiased estimate for the true RV and their scatter provides a sense for the RV uncertainty. However, in reality, some effects may cause the chunk RV to be biased and have a constant offset from the true RV. One of the leading culprits is the error in the wavelength solution of a DSST. As mentioned in the previous section, the measured RV for each chunk at a certain epoch $T_{\rm obs}$ is a relative RV against the DSST taken at epoch T_0, $v_{\star,T_{\rm obs}} - v_{\star,T_0}$. This is because the stellar lines and their Doppler shift in each chunk are modeled by red-shifting the DSST: $F'_\star(\lambda) = F_\star(\lambda \cdot (1+z))$. The wavelength solution for the DSST implies its absolute RV $v_{\star,T_0}$ at T_0. If the wavelength solution for all DSST chunks corresponds to the exact same value of $v_{\star,T_0}$, then measured RVs for all chunks in the observed stellar iodine observation would have the same "zero point". Consequently, any biases or relative errors in the DSST wavelength solution would result in a shifted zero point for that chunk with respect to other chunks. For example, one source of error comes from the fact that the wavelength solution for DSST is derived from neighboring B star + iodine observations, which assumes that the wavelength solution for the orders and pixels remain the same between the B star and the DSST observations. Or, the wavelength solution derived from B star observations may be imperfect. There are also other reasons why the RV zero point of a chunk deviates from the other chunks. Things like persisting CCD effect and certain errors in iodine atlas or DSST can cause biases in RV estimates that contain a constant shift component. In the end, the inconsistent RV zero points will translate into RV scatter as we take the average of all chunk RVs to estimate the RV for one observation. Thus, it is very important to determine and correct for the chunk RV zero point offsets.

Second, not all spectral chunks are equal in terms of the quality of their reported RVs, meaning that we need to take a weighted average. There are several reasons why one chunk would consistently have a larger RV scatter than the other. The most obvious one is difference in the amount of Doppler information contained in the chunks. Some chunks have more and/or deeper stellar/iodine lines, which make them more powerful in accessing the RV information of the star. Some chunks land on the peak of the blaze, which constantly give them more SNR over the chunks down near the bottom. Some chunks may contain stellar lines that have more sensitive response to stellar activity, which would manifest as RV scatter or "jitter". This is not an exhaustive list, but the important thing is that there could be many reasons which we know or do not know, and even for some of the things we do know, we could have no way of estimate how much extra RV scatter it would introduce to the chunk. Therefore, it is most sensible to derive the chunk weights empirically, rather than using any a priori ones.

Overall, the algorithm for vanking works like this:

- It rejects chunks (and observations) with poor performances indicated by, for example, high χ^2_ν values.
- It brings all chunks to having the same RV zero point.
- It computes the weighted average RV for each observation based on the chunk RVs, where the weight for each chunk in each observation is evaluated by how well this chunk behaves in long term (by comparing it with other chunks across all observations). Then the weight is also scaled by how good each observation is, which is evaluated by comparing the behavior of a typical chunk in one observation with the other observations.

To be specific, most of the heavy lifting is done in `vel.pro` and `vdclean.pro`. First `vel.pro` calls `vdcube.pro`, which combines all `vd` structures stored in the `vd` files for all the good observations. A good observation is defined by: (1) median photon counts for all chunks is within a user-defined range; and (2) median χ^2_ν values of all chunks is lower than the user-defined threshold. Otherwise `vdcube.pro` will throw out the bad observation and print out warning messages. Second, the `vd` cube put together by `vdcube.pro` gets passed on to `vdclean.pro`, and `vdclean.pro` performs a series tasks including quality checks, outlier rejections, RV zero point offset adjustment, and finally the computation of chunk weights, i.e., it

1. Throws out chunks with bad DSST or containing no Doppler information (meaning it has a weight of 0 as calculated following the method described in Butler et al. 1996a).

2. Rejects the chunks where the fitter constantly fails to converge (indicated by setting χ^2_ν to 0 or 100 in the code).

3. Rejects the bottom 1% (or other user-defined threshold) of the chunks which have the highest photon-limited RV errors (calculated following Butler et al. 1996a).

4. Rejects the bottom 1% (or other user-defined threshold) of the chunks which have the highest χ^2_ν values.

5. Computes the RV zero point offsets for all chunks and adjust all chunks to have the same zero points. Mathematically, this is done for each chunk by subtracting the offset velocity, which is estimated by the mean velocity of each chunk in all observations, i.e.:

$$\text{offset for chunk } i = \sum_{j=1}^{N} v_{i,j}/N \tag{2.4}$$

   where $v_{i,j}$ means the reported RV for a chunk with index i in observation j, and there are N observations and M chunks in total, so $v_{i,j}$ is a $M \times N$ matrix. This is basically requiring that the mean RV reported by any chunk over all observations is set to zero (or any arbitrary value, since we only care about the RV variation of the star instead of its absolute velocity).

6. Computes chunk weights as the inverse of the estimated RV variance of each chunk i in each observation j, i.e., $w_{i,j} = 1/\sigma^2_{i,j}$, and

$$\sigma_{i,j} = r_j \cdot \sigma_i, \tag{2.5}$$

   where σ_i is defined for each chunk and r_j is defined for each observation as:

$$\sigma_i = \mathit{std}\,\{\Delta_{i,j},\ j = 1, ..., N\} \tag{2.6}$$
$$r_j = \text{median}\,\{|\Delta_{j,i}| \cdot \sigma_i,\ i = 1, ..., M\} \tag{2.7}$$

   where *std* stands for standard deviation, and $\Delta_{i,j}$ is the matrix of velocity differences between $v_{i,j}$ and the median velocity of all chunks in each corresponding observation j:

$$\Delta_{i,j} = v_{i,j} - \widetilde{v}_j \tag{2.8}$$
$$\widetilde{v}_j = \text{median}\,\{v_{i,j},\ i = 1, ..., M\}, \tag{2.9}$$

7. Rejects top 1% of the chunks which have the highest σ_i.

Then `vdclean.pro` passes back the $v_{i,j}$ and the $w_{i,j}$ matrices to `vel.pro`, which rejects the top 1% chunks that have the largest $v_{i,j} - \widetilde{v}_j$ for each observation. Then `vel.pro` computes the weighted mean RV for each observation,

$$v_j = \sum_{i=1}^{M} (v_{i,j} \cdot w_{i,j}) \,/ \sum_{i=1}^{M} w_{i,j}. \tag{2.10}$$

It is worth noting that because $w_{i,j} = \sigma_j \cdot r_j$, this is equivalent to

$$v_j = \sum_{i=1}^{M} (v_{i,j} \cdot \sigma_i) \,/ \sum_{i=1}^{M} \sigma_i \tag{2.11}$$

However, r_j enters the picture when calculating the corresponding RV uncertainty,

$$\sigma_{v_j} = \left(\sqrt{\sum_{i=1}^{M} w_{i,j}.} \right)^{-1} . \tag{2.12}$$

v_j nad σ_{v_j} are then passed back to `jjvank.pro` and stored in the `cf3` structure in the `vst` file for the star.

The algorithm of vanking does not have the most rigorous statistical justification. It was more or less conjured up intuitively and tweaked until it worked, and it works extremely well: the RV scatter is decreased by typically a factor of two or more before and after vanking. It also efficiently eliminates problematic chunks such as the ones with telluric contamination, and as a result, it mitigates the adverse effects caused by the contamination (Section 4.1). Vanking certainly has room for improvement, just like the rest of the code, which is a topic that will be touched on in Chapter 7.