1 Parameter Estimation

A non-dimensional number: $\mathbb{E}\mathrm{u}_e$.

We find the parameters $\mathbf{x}$ that solve the inverse problem $G(\mathbf{x}) = \mathbf{d}$, using a direct search method (*Nelder-Mead*).

$$\min\ \chi^2 = \min\ \sum_{i=1}^{n} \frac{\left(y_d(\mathbf{x})_i - y_G(\mathbf{x})_i\right)^2}{y_G(\mathbf{x})_i}$$

$$\mathbf{x} = \begin{cases} q \\ V_d \\ \sigma \end{cases} \quad \text{subject to constraints} \quad g = \begin{cases} V_d & \pm\ u_{exp} \\ \sigma & \pm\ u_{exp} \\ y_0 & \pm\ u_{exp} \\ t_0 & \pm\ u_{exp} \end{cases}$$

where $y_G(\mathbf{x})$ is a numerical solution of the equation of motion

$$my'' = \frac{1}{2}\rho C_D A_d y'^2 + qE(y) + \frac{1}{4}\frac{Kq^2}{y^2} + \frac{1}{2}E(y)^2\nabla\epsilon$$

In particular, it is impractical to directly measure the droplet net charge q, during a bounce experiement. Our workflow to identify these parameters is as follows:

1. Experimentally vary V_d, σ and capture droplet trajectories using a high-speed camera.
2. Digitize droplet trajectories by using automatic tracking of ellipse-fitted centroids on the thresholded video.
3. Slice droplet trajectories by their bounce minima, and apply a smoothing filter.
4. Maximize the goodness of fit between the experimental trajectories and solutions of the equation of motion, by varying the design vector (which encodes the unkown parameters in the equation of motion) using a direct search optimizer.

Inverse Problems

- We have a dynamical model describing the droplet electro-bounce; we wish to find typical values of the key parameter, droplet net charge q. In general it is not always possible nor practicle to measure model parameters experimentally. They may be hard, expensive, time consuming or perhaps even impossible to measure.
- A mathematical model designed to fit experiemental data so as to explicitly quantify physical parameters of interest.
- Values of model parameters are obtained using parameter estimation techniques aimed at providing a "best fit" to the data.
- Generally involves an interative process to minimize the average difference between the model and the data.

- Evaluating the quality of an inverse model involves a combination of established mathematical techniques as well as intuition and creative insight.
- A good inverse model has: good fit, model parameters are unique, and parameter values consistent with physical inuition or prior knowledge.
- The steps in the process are: 1. Select an appropriate mathematical model from theory. 2. Define a "figure of merit" function which measures agreement between the data and model for a given set of parameters. 3. Adjust model parameters to get a "best fit". 4. Evaluate "goodness of fit" to data, which tends never to be perfect due to experimental noise. 5. Estimate accuracy fo the best-fit parameter values (provide confidence intervals and determine uniqueness). 6. Determine whether a much better fit is possible (which is difficult of ruggest respose surfaces; F-test can be used for comparing models of different complexity).
- Maximum likelihood estimation: not "what is the probability that my set of model parameters is correct?" but rather "given my set of model parameters, what is the probability that this data set occurred (what is the likelihood of the parameters given the data)?"
- [notes on χ^2 goodness-of-fit]
- Model parameter estimation is the process of indirect determination of unknown model parameters from measurements of experiemental data.
- This is mathemetically challenging [ref]. There are practical challenges as well, as experimental data tends to be noisey and sparse.
- In the most familliar sense determining model parameters is done using derivative based methods, essentially by estimating derivatives by finite differences (Euler's method) and fitting paramters using linear regression. However there are approaches to fitting any arbitrary function.
- Suppose we have a model $G(\mathbf{m})$, with a vector of parameters $\mathbf{m}$, and set of perfect noiseless observations $\mathbf{d}$, we expect there to exist a relationship

$$G(\mathbf{m}) = \mathbf{d}$$

  where the operator G might be an ODE.
- Suppose we have a model given by the ODE

$$\frac{dy}{dt} = f(t, \mathbf{y}; \mathbf{p}), \mathbf{y} \in \mathbb{R}^n, \mathbf{f} \in \mathbb{R}^n,$$

  where $\mathbf{p} \in \mathbb{R}^m$ is the vector of parameters, and a collection of measurements of experiemental data

$$(t_1, \mathbf{y_1}), (t_2, \mathbf{y_2}), ..., (t_k, \mathbf{y_k}).$$

  We wish to minimize an objection function which is the mean square error (or any goodness of fit figure of merit in general) between the model output and experimental data:

$$\text{obj}(\mathbf{p}) = \sum_{j=1}^{k} |\mathbf{y}(t_j; \mathbf{p}) - \mathbf{y_j}|^2,$$

where $|.|$ is the Euclidian norm. Therefore the parameter estimation problem is a variety of optimization problem.

- The process of fitting a function, defined by a collection of parameters, to a data set is called the discrete inverse, or parameter estimation problem (as opposed to the *forward problem* to find $\mathbf{d}$ given $\mathbf{m}$ and $G(\mathbf{m})$).

Optimization

Most generally the we state the constrained problem as

$$\begin{array}{lll} \text{minimize:} & F(\mathbf{x}) & \text{objective function} \\ \text{subject to:} & & \\ & g_j(\mathbf{x}) \leq 0 & \text{inequality constraints} \\ & h_k(\mathbf{x}) = 0 & \text{equality constraints} \end{array}$$

$$\text{where } \mathbf{x} = \begin{cases} x_1 \\ x_2 \\ \vdots \\ x_n \end{cases} \quad \text{design variables}$$

Our specific optimization problem is non-convex, mixed discrete-continuously black-box (noisy), which is essentially the worst the worst case scenario. While in principle a gradient-based optimizer (such as ...) could be used by using finite-differences to obtain approximate gradients of the χ^2 objective funtion, in practice doing so is problematic becasue the signal-to-noise ratio of the objective function scales like $\mathcal{O}(f)$ for $\frac{df}{dt}$ and $\mathcal{O}(f^2)$ for $\frac{d^2f}{ft^2}$ which will cause problems with our Hessians and Javobians respectively [Refs 5, 14, 49, 95.]. As a further practical matter, given the relatively expensive function-calls (which requires solving a stiff, non-linear ODE) gradient-free approaches tend to offer better performence regardless.

[notes on gradient free optimization, Nelder-mead, sci-py ref]

- We use the *Nelder-Mead* algorithem to minimize the χ^2 goodness-of-fit. *Nelder-Mead*, sometimes called simplex-search or downhill-simplex, is a derivative-free direct search, LP method.
- *Nelder-Mead* is well-established.
- A hueristic search method, with no guarantee of optimal solutions.
- Is based on the concept of a simplex, which is a special polytrop of $N+1$ vertices in N diemsions.
- Is derivative-free: does not use numerical or analytic gradients.
- Is implimented in the *Scipy* package for Scientific computing in Python.
- *Nelder-Mead* is very fast, but generally suitable for low-dimensional problems. Additionally convergence of the alogrithem is highly sensitive to the initial guess design vector x_0.

We do not have *a priori* information regarding existence of true globality for our objective function. Perhaps more importantly, given the degrees-of-freedom in the parameter space there may be a multiplicity of local extrema of the χ^2 hyper-response surface. *Nelder-Mead* is not a global optimizer, though there are varients which use sequential local searches with probablistic restarts to achieve globality. However global optimization usually comes at a tremendous computational cost.

That we are capabible of fitting any arbitrary model to a dataset given sufficient degrees of freedom in our parameters is admittedly a disconcerting issue, and is manifested in the locality of the extrema of the χ^2 response surface. However, some of the inverse model parameters are constrained by our experimental observations of them and their associated measurement uncertainties. This, we hope, makes the spectre of an overfitted model less frightening, but does convert our uncontrained optimization problem to an constrained one which raises special difficulties of its own.

The *Nelder-Mead* direct search method cannot be used with explicity constrained problems. However, there are various implicit approaches to (approximately) solving general constrained problems using uncontrained algorithems. Generally, this is achieved by domain transformations or the use of penalty functions. By the addition of a penalty function which depends in some way on the values of the constraints to the objective function, we minimize a pseudo-objective function where the infeasibility of the constraints is minimized simultaneously to the objective function.

There are various penalty function schemes. We use an Exterior Penalty Function as a simple way of converting converting the constrained problem into an unconstrained one. These are are especially useful in cases where the constraints are not "hard" in the sense that they need to be satisfied precisely. General penalty functions, which are sequential unconstrained minimization techniques, reformulate the general constrained problem as the pseudo-objective function given by

$$\phi(\mathbf{x}, r_p) = F(\mathbf{x}) + r_p P(\mathbf{x})$$

where $P(\mathbf{x})$ the penalty function, is given by

$$P(\mathbf{x}) = \sum_{j=1}^{m} \left\{ \max\left[0, g_j(\mathbf{x})\right] \right\}^2 + \sum_{k=1}^{l} \left[h_k(\mathbf{x}) \right]^2 . \tag{1}$$

We see from Equation 1 that there is no penalty if the constraints $g_j(\mathbf{x})$, and $h_k(\mathbf{x})$ are satisfied.

The Exterior Penalty Function specifically (and all Penalty Function approaches in general) do have several drawbacks. Namely these include the possibility of the objective function being undefined outside of the set of feasible solutions. Additionaly, by naively "encouraging" feasibility of the solution using large values of the penalty parameter, r_p, we will tend to ill-condition the unconstrained formulation of the problem. [notes on choice of optimizer parameters, initial guesses, convergence, error estimates]. Using penalty function methods tend to exascerbate the sensitivity of *Nelder-Mead* solutions to the choice of intial guess vector [ref].

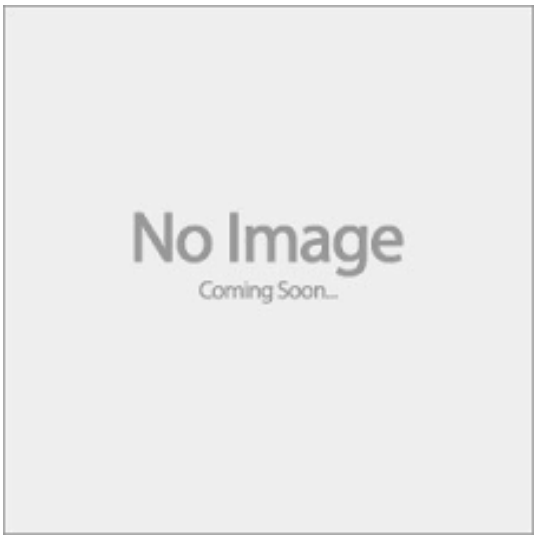

Figure 1: Filtered or fitted droplet trajectories compared to raw data.

Smoothing

In trying to recover the kinematic variables from the droplet trajectories we encounter a difficulty in the noise of the position data:

Error sources include misalignment of the camera, perspective due to objects (subject or reference scale) being out of the photographic plane, precision limits in digitization. Some of these errors are systematic in origin and introduce consistent biases into the data (e.g. coherent spectral sources, rather than truly stochastic noise). These systematic sources of error include inaccurate scales among others. Data smoothing tends not to help with the systematic errors in that they are usually of lower frequency than the signal (and here we are trying specifically to filter high frequency noise). Random errors, by contrast, are assumed to have a Gaussian distribution (by the central limit theorem), and are independent of the signal (which inherently results from a deterministic process).

In lieu of global polynomial fits we can attempt at finite differencing the data after the application of smoothing filters. Some of these approaches are global and/or recursive whereas other approaches are applied locally. Each approach has various natural advantages and disadvantages. A number of approaches were implimented in *Python* or imported from the scientific computing package *SciPy*[ref] and compared for a test set of trajectory data; these methods include convolution with the derivative of a Gaussian kernel, Wiener filtering, smoothing splines, Butterworth low-pass filtering, total-variance regularization, and Savitsky-Golay filtering. Qualitatively comparing these smoothing methods we find that we loose too many data points in the smoothing process, the variance remains large (especially in higher derivatives), large amplitudes are overly smoothed by repeated filtering passes, or there are significant end effects for most of these methods. A comparison of these smoothing approaches on a representative trajectory data set are shown in Figure 1. The power spectra for the same data are compared for these methods in Figure 2.

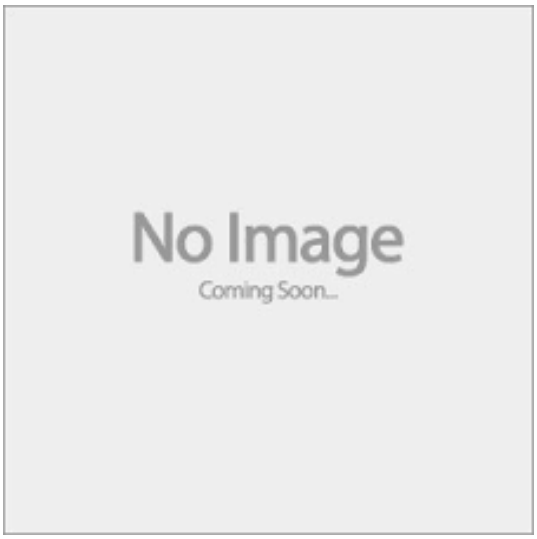

Figure 2: Power spectra of the filter methods compared.

Savitsky-Golay filtering seems to produce the smoothest derivatives without overfitting of otherwise misrepresenting the underlying signal.

Numerical Solution to the Equation of Motion

The equation of motion behaves stiffly due to the large disparity in Coulombic, image charge, and dielectrophoretic lenghtscales. We integrate it numerically using the `odeint` *Scipy* module. This is a shake-and-bake Python wrapper for the vernerable 1982 *netlib ODEPACK* library double-precision `lsoda` (Livermore Solver for Ordinary Differential equations with Automatic method switching for stiff and nonstiff problems) integrator [ref]. The function switches between Adams (nonstiff) and Backwards Differentiation Formulas (BDF, stiff) according to the dyanmic value of a set of stiffness eigenvalues.