Need to update because changed to stiff solver ode15s because was having jaggedness problems with P class with ode45 for some choices of parameters because of orders of magnitudes. Read "Solve stiff ODEs" matlab documentation in order to argue we needed to use this. The MATLAB ode45 solver is the first choice for solving non-stiff ODE's, meaning there is no issue with solution components varying drastically on different time scales and doesn't require small steps for accurate results, and is a medium order method, meaning provides medium level accuracy. Further reading on stiff systems: 572 notes chapter 9. Explain here the options used with ode45 in code. Although the solver returns the solution evaluated at the given tspan points, it does not step only to each point specified in tspan to compute the solution and instead uses it's own internal steps. It uses an explicit Runge-Kutta (4,5) formula, in which each iteration only needs information on the solution at the preceding time point. See Dormand, J. R. and P. J. Prince, "A family of embedded Runge-Kutta formulae," J. Comp. Appl. Math., Vol. 6, 1980, pp. 19-26 for more details. Further reading on stiff systems: 572 notes chapter 9. Local error is computed at each time step which gives the error in the state values. The relative tolerance gives the error relative to the size of the respective state with the default value being $1e^{-3}$; as the state being measured approaches zero, the absolute tolerance gives the acceptable error value which depends on the relative tolerance value. Overall, the error for the j^{th} state must be $\leq max(rtol * |x_j|, atol_j)$. Put in more details about this, if needed.

Need to put in details about general ordinary least squares from some source. Maybe use MatLab documentation and/or paper/textbook.

The MATLAB MultiStart solver in the Global Optimization Toolbox utilizes a local solver, such as fmincon in our case, at multiple points in order to find multiple local minima of a given problem. First, MultiStart checks the problem inputs by running fmincon once to make sure they are valid. Understand better...do a test run with "incorrect problem inputs." Then, given the user-supplied starting point and a desired n starting points total, MultiStart generates $n-1$ starting points and it chooses starting points based on a uniform distribution (i.e. equal probability for each point to be chosen) unless is means uniformly distributed points? within the bounds provided. MultiStart calls "list" which returns points based on the RandomStartPointSet object containing parameters for generating points. Need to understand more?

Next, it runs fmincon with these starting points and runs until the maximum time allotted by the user for each iteration; if it exceeds that many seconds, then it exits the iteration without giving a solution. The MATLAB fmincon solver is a local solver that finds the minimum of a constrained, nonlinear, multivariable function; in our case, we give lower and upper bounds on parameters. The local solver finds the local minimum in the basin of attraction of each starting point, and therefore, MultiStart searches through multiple basins of attraction for the optimum. The negative gradient of the goal function to minimize points in the direction where the function most quickly decreases, and solving the steepest descent equation for a single variable function

$$\frac{dx(t)}{dt} = -\nabla f(x(t))$$

results in a path $x(t)$ that goes to a local minimum as time increases. Understand more. The iterations of the method take the form

$$x_{i+1} = x_i - \gamma \nabla f(x_i).$$

The set of initial values that lead to the same solution is known as the basin of attraction. The default optimization algorithm of fmincon is the interior point algorithm which sets components of starting points to be strictly between the bounds if they initially are not. See fmincon Interior Point Algorithm if need more details.

Confused how interior points algorithm comes in and what it's for.

First order optimality measures how close a point x is to optimal, and it is necessary to be 0 at a minimum, but the solver requires it to be less than an optimality tolerance with a default of $1e^{-6}$. The solver returns the solution, the value of the objective function at the solution, and an exitflag in which a positive value of the latter means that a local minimum has been found. I believe this is a stoping criteria for fmincon which is equivalent to the gradient being 0 (or very small) which is necessary for a minimum (but not sufficient), look more at tolerances/stopping criteria in matlab documentation

MultiStart runs all start points and stops when it runs out of start points, or it's total run time exceeds the maximum time allotted. Solutions are considered identical by the solver if they are within a default $1e^{-6}$ relative distance from one another and their objective function values are also within a default $1e^{-6}$ relative difference from one another. Once MultiStart has stopped, it sorts the local solutions based on the value of the objective function from lowest to highest. It then loops over all of the local solutions j starting with the best solution to find all solutions k such that:

$$|Fval(k) - Fval(j)| \leq FunctionTolerance * max(1, |Fval(j)|)$$
$$|x(k) - x(j)| \leq XTolerance * max(1, |x(j)|).$$

It records j, $Fval(j)$ and...super confused here about the k idea that comes next. The idea of this is that if have a local minimum, 4 starting points, for example, may end up at the same local min so this looping throws away 3 of them since they all are the same answer and this reduces the number of entries in GlobalOptimSolution; also, if distance between two "local mins" is within 10^{-6}, they are viewed as the same local min, etc. This creates one entry in the GlobalOptimSolution object vector. Therefore, this object contains the information on where the local minimum is, the corresponding value of the objective function, and the starting points that led to the minimum. Once complete, a summary is given that provides information on the number of fmin con runs that converged, didn't converge, or had errors.

Although GlobalSearch has a similar method to finding global minima and is most efficient, MultiStart searches more thoroughly for a global minimum. Give any other reasons why choose ms over gs.

Put in sources (commented out right now).