Ordinary Differential Equations

Part 4

Massimo Ricotti

ricotti@astro.umd.edu

University of Maryland

Two-point Boundary Value Problems

- *NRiC* §17.
- BCs specified at two or more points, e.g., start and end.
- For IVP, just integrate away.
- For 2-pt BVP, must make a free choice of unknown BVs at initial point, then integrate away. But solution will almost certainly *not* satisfy other BCs at end.
- Strategy: Use information about how much the other BVs "missed" to iteratively improve initial guess.

  $\Longrightarrow$ Techniques are all iterative (and expensive).

Notation

- Denote standard system as:

$$\frac{dy_i(x)}{dx} = g_i(x, y_1, ..., y_N) \qquad i = 1, ..., N.$$

- At x_1, the solution is supposed to satisfy:

$$B_{1j}(x, y_1, ..., y_N) = 0 \qquad j = 1, ..., n_1.$$

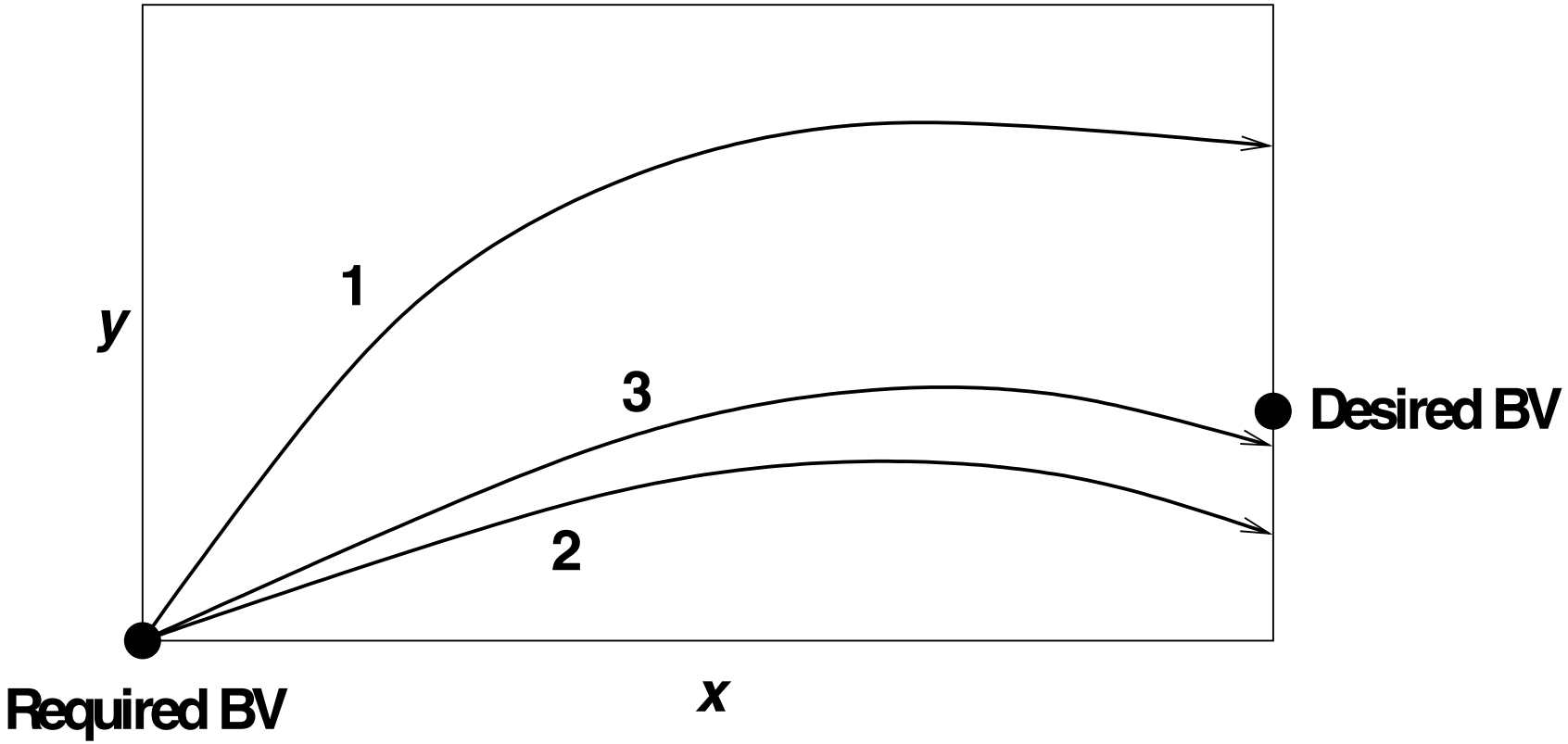
- At x_2, it is supposed to satisfy:

$$B_{2k}(x, y_1, ..., y_N) = 0 \qquad k = 1, ..., n_2,$$

where $n_2 = N - n_1$.

Two Basic Techniques: Shooting method

1. Begin at x_1.
2. Guess values for free BCs (n_2 values).
3. Integrate as IVP to x_2.
4. Adjust n_2 guesses to get closer to BVs at x_2.

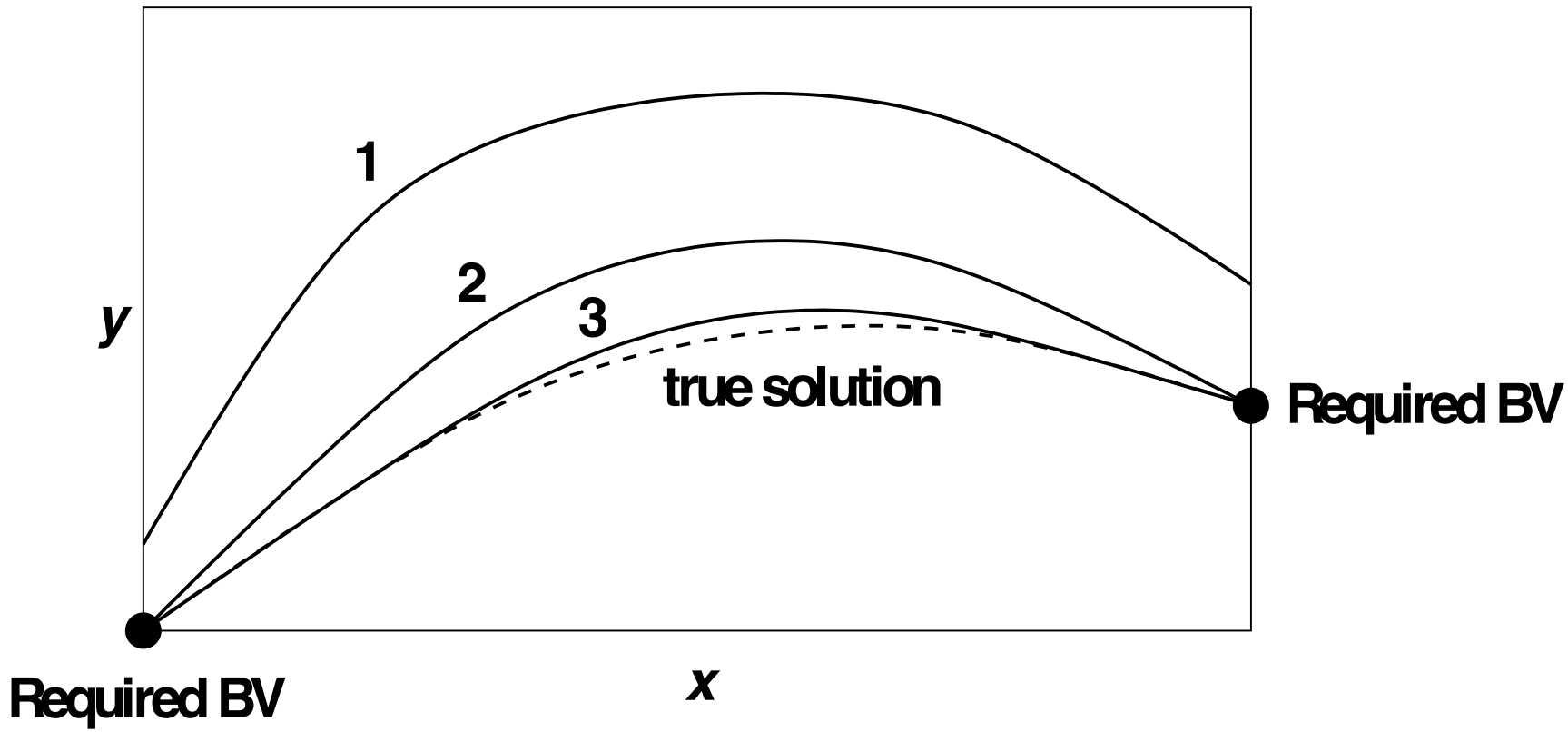


- Heart of technique: system of iteratively improving guesses.

  $\Longrightarrow$ Multi-D root finding.

Two Basic Techniques: Relaxation method

1. Replace ODEs by finite-difference equations on mesh from x_1 to x_2.
2. Guess solution on this mesh.
3. Mathematically, FDEs are just algebraic relations between unknowns. Use iterative technique to relax this solution to true solution.



- Relaxation very powerful for smooth solutions, or ODEs that must be solved many times with different parameter values. Also good when ODEs have extraneous solutions, i.e., stiff equations.
- *NRiC*: “Shoot first, relax later.”

2-pt BVP: Shooting Method

1. At x_1, must specify N starting values for y_i, $i = 1, ..., N$ (in NR user provided function `load()`).
   - n_1 values given by BC at x_1.
   - $\therefore n_2 = N - n_1$ values can be freely chosen.
2. Represent the free values as a vector $\mathbf{V}$ of dimension n_2 (actually, $\mathbf{V}$ represents schematically any parameter value that specifies unknown BVs).
3. Now integrate to x_2.
4. Define "discrepancy vector" $\mathbf{F}$ of dimension n_2 (in NR, user provided function `score()`), where

$$F_k = B_{2k}(x_2, \mathbf{y}) \qquad k = 1, ..., n_2.$$

   - We want to find $\mathbf{V}$ that zeroes $\mathbf{F}$.

5 Solve n_2 linear equations:

$$\mathbf{J}\,\delta\mathbf{V} = -\mathbf{F},$$

where $J_{ij} \equiv \partial F_i/\partial V_j$ is the Jacobian matrix.

- This is the globally convergent Newton's method (*NRiC* §9.7).

6 Then $\mathbf{V}_{\rm new} = \mathbf{V}_{\rm old} + \delta\mathbf{V}$.

7 Use $\mathbf{V}_{\rm new}$ to solve ODEs again as IVP, recompute $\mathbf{F}$, and iterate again until $|\mathbf{F}| < \varepsilon$, the convergence criterion.

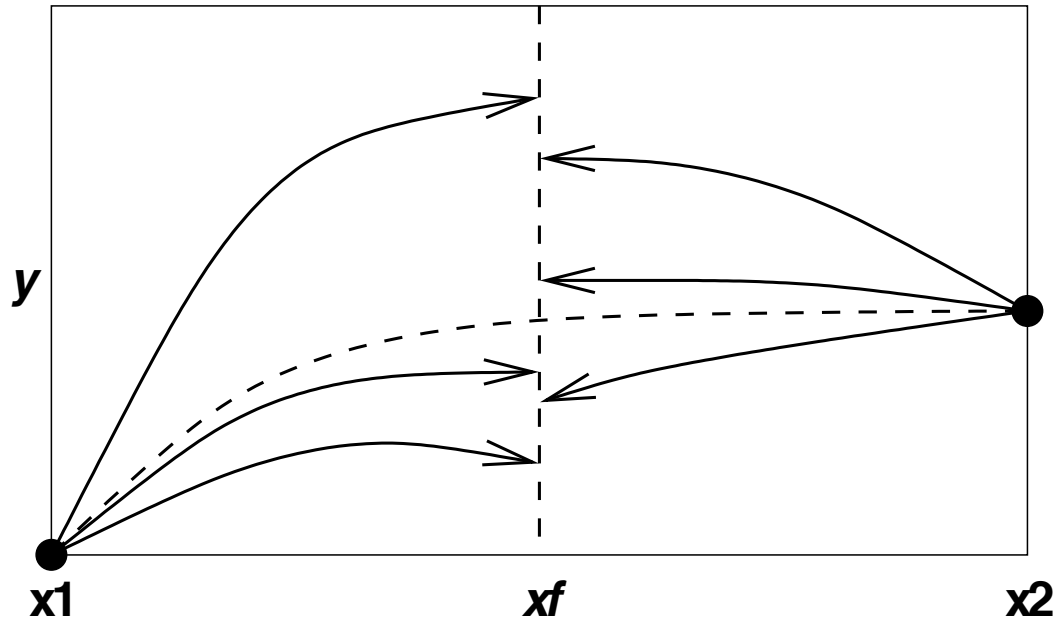
- Cannot compute Jacobian analytically. Instead evaluate differences numerically, i.e.,

$$\frac{\partial F_i}{\partial V_j} \simeq \frac{F_i(V_1, ..., V_j + \Delta V_j, ..., V_{n_2}) - F_i(V_1, ..., V_j, ..., V_{n_2})}{\Delta V_j},$$

  i.e., solve IVP n_2 times varying each component of $\mathbf{V}$ by $\Delta\mathbf{V}$ each time to build up Jacobian (recall $F_i(V_1, ..., V_{n_2})$ already computed in step 4).

- Overall procedure requires $n_2 + 1$ solutions to ODEs per iteration.
- For linear systems, one iteration is enough.
- For nonlinear systems, many (say M) iterations may be required to converge $\Longrightarrow M \times (n_2 + 1)$ solutions of ODEs!
- $\therefore$ need efficient integrator... (*NRiC* routine `shoot()` uses `odeint()` and is called by `newt()`).

- NOTE: Can also shoot to a fitting point between x_1 and x_2 (*NRiC* §17.2). Useful for singular BC(s) and/or domain point(s).



```
#define NRANSI
#include "nrutil.h"
#define EPS 1.0e-6

extern int nvar;
extern float x1,x2;

int kmax,kount;
float *xp,**yp,dxsav;

void shoot(int n, float v[], float f[])
{
void derivs(float x, float y[], float dydx[]);
void load(float x1, float v[], float y[]);
void odeint(float ystart[], int nvar, float x1, float x2,
float eps, float h1, float hmin, int *nok, int *nbad,
void (*derivs)(float, float [], float []),
void (*rkqs)(float [], float [], int, float *, float, float,
float [], float *, float *, void (*)(float, float [], float [])));
void rkqs(float y[], float dydx[], int n, float *x,
float htry, float eps, float yscal[], float *hdid, float *hnext,
void (*derivs)(float, float [], float []));
void score(float xf, float y[], float f[]);
int nbad,nok;
float h1,hmin=0.0,*y;

y=vector(1,nvar);
kmax=0;
h1=(x2-x1)/100.0;
load(x1,v,y);
odeint(y,nvar,x1,x2,EPS,h1,hmin,&nok,&nbad,derivs,rkqs);
score(x2,y,f);
free_vector(y,1,nvar);
}
#undef EPS
#undef NRANSI
/* (C) Copr. 1986-92 Numerical Recipes Software ?421.1-9. */
```

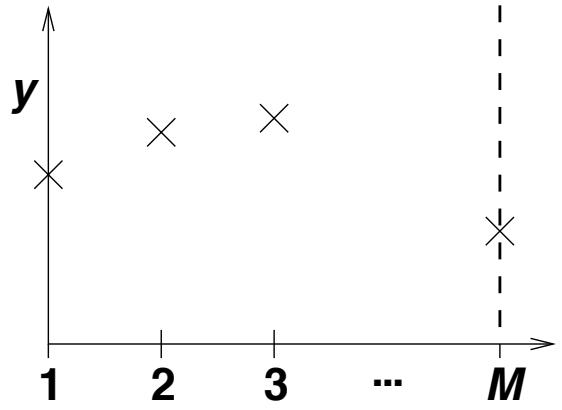
2-pt BVP: Relaxation Method

Procedure:

1. Replace ODE (e.g., $dy/dx = g(x, y)$) with FDE on a grid:[a]

$$y_k - y_{k-1} - (x_k - x_{k-1})\, g\left[\tfrac{1}{2}(x_k + x_{k-1}), \tfrac{1}{2}(y_k + y_{k-1})\right] = 0.$$

- Here x_k and y_k are discrete values of independent and dependent variables at "mesh points."



[a] This is not a unique representation.

- For M mesh points and N coupled equations, have $M \times N$ y_k's to solve for.

2 Guess starting values for all y_k.

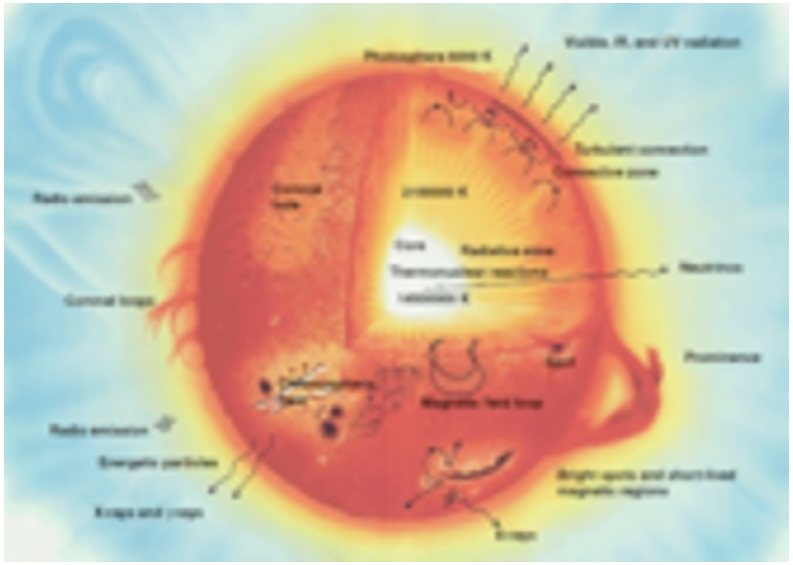
3 Iteratively improve ("relax") solution using N-R. The correct solution is obtained when the boundary conditions are satisfied and the difference equations between grid points, like the one above, are zeroed to the desired accuracy.

- Need to solve a matrix equation each iteration, where the matrix is $(M \times N) \times (M \times N)$ in size, but is also block diagonal (for which efficient solving algorithms exist). The block diagonal form comes from first-order Taylor series expansions of the FDEs about each pair of grid points.[a]
- Choice of grid points is an important issue and leads to adaptive mesh strategies in modern solvers.

---

[a] Each interior point supplies a block of N equations coupling $2N$ corrections to the solution variables at points k, $k-1$. The boundary conditions supply smaller blocks, $n_1 \times N$ and $n_2 \times N$.

Example: Stellar Structure



- Numerical methods for 2-pt BVPs largely developed by astronomers seeking to solve equations of stellar structure.
  - Form a system of four coupled ODEs.

Equations of stellar structure

1. Consider spherical shell, thickness dr, distance r from origin. Then $dM = 4\pi r^2 dr\rho$, or,

$$\frac{dM}{dr} = 4\pi r^2 \rho.$$

2. Hydrostatic equilibrium $\Longrightarrow$ net force on shell is zero. $\therefore -\nabla_r P - \rho g = 0$, where g = gravitational acceleration per unit mass = GM/r^2, or,

$$\frac{dP}{dr} = -\frac{GM}{r^2}\rho.$$

(To derive, note upward force on shell per unit area = $P(r) - P(r + \Delta r) = -\Delta P$ must equal downward force per unit area = $[GM(r)/r^2](M_{\rm shell}/4\pi r^2) = (GM/r^2)\rho\, dr$.)

3. Let ε = energy generation rate/unit mass. Then energy transport rate through shell $\Delta L = L(r + \Delta r) - L(r)$ must equal energy generation rate $4\pi r^2 dr\, \varepsilon\rho$, where L = luminosity, or,

$$\frac{dL}{dr} = 4\pi r^2 \varepsilon\rho.$$

4. Finally, there is a relationship between luminosity through shell, "thermal conductivity" across shell, and temperature gradient (transport dominant in white dwarfs):

$$\frac{dT}{dr} \propto -\frac{L}{\kappa},$$

where,

$$\begin{aligned} \text{higher } \kappa &\rightarrow \text{lower } T \text{ gradient;} \\ \text{higher } L &\rightarrow \text{higher } T \text{ gradient.} \end{aligned}$$

- This equation harder to derive since it depends on energy transport mechanism and convective stability that differs from star to star and with depth inside a star. Eg, for radiative transport:

$$\frac{dT}{dr} \propto -\frac{\alpha \rho L}{\sigma T^3},$$

where α is the gas opacity.

- This gives 4 ODEs in 7 unknowns ($M, \rho, P, L, T, \varepsilon, \kappa$).
- Need 3 constitutive relations:
  1. $P = P(\rho, T)$ — equation of state.
  2. $\varepsilon = \varepsilon(\rho, T)$ — nuclear energy generation rate.
  3. $\kappa = \kappa(\rho, T)$ — thermal conductivity.

- Boundary conditions (need 4):
  - At $r = 0$: $M = 0$, $L = 0$.
  - At $r = R_\star$ ($M = M_\star$): $P = 0$, $\rho = 0$ ($\Rightarrow T = 0$).
- This is a classic 2-pt BVP.
- First techniques developed based on shooting method.
- Singularity at $r = 0 \Longrightarrow$ must fit at an intermediate point: Schwarzschild Scheme (e.g., Schwarzchild, *Structure and Evolution of the Stars*).
- Modern stellar structure codes use relaxation method with adaptive mesh (e.g., P. Eggleton, *MNRAS*, **151**, 351 (1971)).

Polytropes

- Can illustrate technique with calculation of structure of polytropes.
- Assume there is no energy generation anywhere inside ($\varepsilon \equiv 0$), e.g., white dwarf or neutron star.
- Assume EOS of form $P = k\rho^{(n+1)/n}$ (no T dependence).
  - E.g., EOS for monatomic gas (such as degenerate electron gas) is:
    $P \propto \rho^{5/3}$ if non-relativistic ($n = 3/2$);
    $P \propto \rho^{4/3}$ if relativistic ($n = 3$).
- This form is called a polytropic EOS. n is polytropic index.

- It is convenient to rewrite $\rho = \rho_c \theta^n$. Then the first stellar structure equation becomes

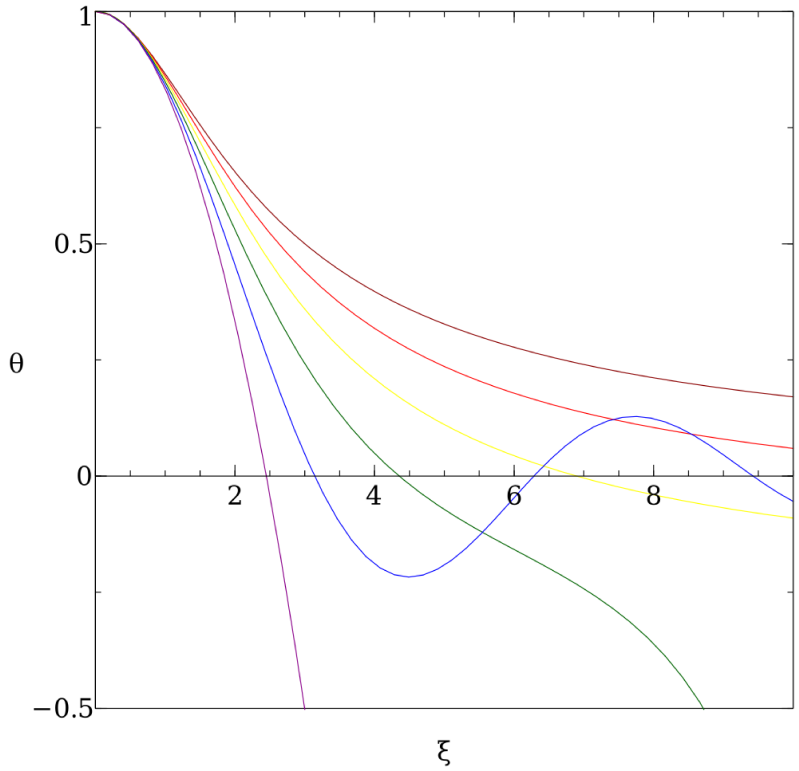
$$\frac{d\mathcal{M}}{d\mathcal{R}} = \mathcal{R}^2 \theta^n,$$

and the second becomes

$$\frac{d\theta}{d\mathcal{R}} = -\frac{\mathcal{M}}{\mathcal{R}^2},$$

where $\mathcal{R} \equiv r/r_0$, $\mathcal{M} \equiv M/M_0$, with $M_0 = 4\pi r_0^3 \rho_c$ and $r_0^2 = (n+1)k\rho_c^{1/n-1}/4\pi G$.

- These are the "Lane-Emden Equations."
- This is a system of 2 ODEs with BC $\mathcal{M} = 0$ at $\mathcal{R} = 0$ and $\theta = 0$ at $\mathcal{R} = R_\star/r_0$.
  - If we have a desired $R_\star$ known in advance (or, equivalently, $M_\star$), then we can set $\theta = 1$ at $\mathcal{R} = 0$, integrate and find $\mathcal{R}$ or $\mathcal{M}$ where $\theta = 0$. The mass or radius wanted at $\theta = 0$ sets ρ_c.



- Solutions of Lane Emden equation for $n = 0, 1, 2, 3, 4, 5$.