Scheid Table of Contents PhL 12.3.04

This is both a table of contents and a brief summary of what happens in each chapter. If you want to know more about a chapter, read Scheid's introduction at the start of the chapter. Or see my own 75 pages of notes, some of which are summaries of things. This book contains an *awesome* amount of information!

1. What is numerical analysis. 1

Look for errors in the data *and* in the algorithm.

2. The Collocation Polynomial. 10

There exists poly p(x) matching y(x) at N equally spaced points.
The **Division Algorithm** reduces poly by one degree (**Remainder Theorem**)
The **Factor Theorem**: p(r) = 0 => (x-r) is a factor of p(x).
Poly p(x) degree n has at most n real zeros (but n complex ones).
Error formula $y(x) - p(x) = \pi(x) * y^{(n+1)}(\zeta) / (n+1)!$
Use of **Rolle's Theorem**, and the **Mean Value Theorem** to get the above.

3. Finite Differences 15

The **forward differences** like $\Delta y_k = y_{k+1} - y_k$, $\Delta^2 y_k = y_{k+2} - 2y_{k+1} + y_k$
List of various **calculus-like rules** for differences, like $\Delta(y_k z_k) = y_k \Delta z_k + z_k \Delta y_k$
Analogs of $\delta(x)$ and $\delta'(x)$ in discrete world.
Notion of a **triangle table** to compute differences from initial y_k

4. Factorial polynomials 22

Example: $y_k = k^{(3)} = k(k-1)(k-2) = k^3 - 3k^2 + 2k = k! / (k-3)!$
Coefficients are **Stirling's first kind numbers**
Reverse solve $k^3 = k^{(1)} + 3k^{(2)} + k^{(3)}$, coeffs are **Stirling's second kind numbers**.
Main usefulness is this **calculus-like** fact: $\Delta k^{(n)} = n\, k^{(n-1)}$, can write as difference.

5. Summation 30

If you can find $\Delta y_k = z_k$, then $\Sigma z_k = y_n - y_1$, the **telescoping sum idea**
Since $\Delta k^{(n)} = n\, k^{(n-1)}$, and $k^n = \Sigma\, a_i\, k^{(i)}$, can do lots of poly sums.
Since $k^{(-n)} = 1/\text{poly}(k)$, can sum inverse polys as well.
Can do **summation by parts.**

6. The Newton (forward) Formula (for the collocation polynomial, equal spaced y_i only) 34

It is $p(k) = \Sigma^n [\, k^{(i)}/i! \,] \Delta^i y_0$ = poly of degree n which matches n+1 points y_k
Discussed in chapter 2, but finally constructed and presented here!

7. Operators and Collocation Polynomials 38

Raising operator $E\, y_k = y_{k+1}$ or fractional $E^x\, y_k = y_{k+x}$
Backward difference operator $\nabla y_k = y_k - y_{k-1}$
Central difference operator $\delta y_k = y_{k+1/2} - y_{k-1/2}$
Mean operator $\mu\, y_k = (y_{k+1/2} + y_{k-1/2}) / 2$
Relations between operators.

Use of operators in the abstract to get interesting results
Many other ways to write the *same* Newton collocation formula:
Newton backward, Gauss forward and backward, Stirling, Everett, Bessel
The are reorderings of the terms, and have different **zigzag** paths through diff data, p 50.

8. Unequally spaced arguments (collocation formula for) 53
It is the **Lagrange** formula $p(x) = \Sigma L_i(x)\, y_i$ where $L_i(x)$ = **Lagrange multiplier**
The **Determinant Form** and **Aitken's Method** are other ways to write it.
Note that $L_i(x_k) = \delta_{ik}$.

9. Divided differences (*another way* to write the unequal spaced collocation poly) 58
Here we repeat the *equal spacing* formula from chapter 6:
$p(x) = y_0 + (x-x_0)\ \{\Delta^1 y_0/h\} + (x - x_0)(x - x_1)\ \{\Delta^2 y_0/2h^2\} + (x - x_0)(x - x_1)(x - x_2)\{\Delta^3 y_0/3!\ h^3\} + ...$
Here is the generalization to unequal spacing: (called Newton's divided difference formula)
$p(x) = y_0 + (x-x_0)\ \{y(x_0,x_1)\} + (x - x_0)(x - x_1)\ \{y(x_0,x_1,x_2)\} + (x - x_0)(x - x_1)(x - x_2)\{y(x_0,x_1,x_2,x_3)\} + ...$
The y things are the "divided differences", fancier versions of the normal differences.
The error formula in Chapter 2 is now: $y(n+1)(\zeta) / (n+1)! = y(x,x_0,x_1,x_2 \ldots x_n)$
Can build a triangle table of divided differences same was

10. Osculating Polynomials 65
In chapter 8 we have unequal spaced poly of degree n to match n+1 points y_i.
Here we get poly of degree 2n+1 to match n+1 points y_i AND n+1 slopes y'_i.
It is **Hermite's Formula** involving the Ui(x) and Vi(x) functions

11. The Taylor Polynomial 70
The series to order N osculates to order N, but only at once point.
Lagrange Error Formula for the omitted tail.
Operators D, $\mathbf{e^D = 1 + \Delta}$ and D^{-1} integral operator, etc.
The **Euler Transformation** to reorder and accelerate a series
The **Bernoulli numbers** (defined only, will get used later)
The **Euler-Maclaurin transformation** for series = integral + corrections
Taylor expand $y(x) = (1 + x)^p$ and you get the **Binomial Expansion**

12. Interpolation and Prediction 79
Use the collocation poly in one of its forms to interpolate data in a table or experiment.
The **Everett** version is popular and contains only even powers of δ^2
Prediction means interpolation off the end of your data.

13. Numeric Differentiation 97
A dangerous thing to do if noise on your data.
Idea is to fit data with the collocation poly p(x) in some form, then diff that.
Better method given later where you first fit the data least squares say then diff.

14. Numeric Integration (with equally spaced sample points) 107
Much better thing to do numerically than differentiation.
Fit curve y(x) with collocation poly p(x) in some form, then integrate that term by term

The **trapezoidal rule** uses linear fit for each piece of curve.
Simpson's rule uses quadratic fit for each piece of curve.
Fit curve with Newton forward (fitting n+1 points), get **Newton-Cotes formulas**.
Romberg's Method (iterative algorithm to do numerical integral)
In general, error doing equal-spaced numeric integration looks like $\sim y^{(n)}(\theta)$

15. Gaussian Integration (unequal spacing, aka **Gaussian Quadrature**) 125
This chapter was my main motivation for reading Scheid's book.
Write approx integral using Chap 10's **Hermite interpolation**, get $\Sigma A_i y_i + \Sigma B_i y_i'$ for RHS.
Result exact for poly y(x) up to 2n-1 degree if use n points; error is $\sim y^{(2n)}(\theta)$, very good.
Note that *any* points n points x_i can be used (so far) for the $y_i = y(x_i)$
You can cause all $B_i = 0$ if you choose x_i as zeros of orthogonal polynomial of order n.
The type of ortho poly needed is controlled by weight w(x) and endpoints a,b of the integral.
The A_i coefficients are independent of y_i, determined by your ortho poly.
Common weight/polys/ab are Legendre, Laguerre, Hermite, Chebyshev, Lobatto.
For example, you might do "Gauss-Laguerre" integration.
The "general theory" is not presented, see other sources. Legendre special case is done.
See notes " the Main Logic of Gaussian quadrature"

16. Singular Integrals 151
Gaussian integration may fail badly for an integral that is singular in the interval.
There are certain tricks that can help
Ignore it and maybe it works anyway.
Subtract out the singular part to get two integrals, and you know how to do the sing one.
Change variables, most popular choice.
Use partial differentiation with respect to a **parameter** α.
Jim and others have written papers in how to handle certain singular looking integrals.

17. Sums and Series 156
Telescoping can do a LOT of things for you in doing sums.
The **Alternating Series Theorem**
Euler-Maclaurin Formula (not same as E-MTransformation)
The **subtraction** method
Summing power-related terms and **Bernoulli polynomials and numbers**.
The **Wallace Products**
Asymptotic Series and **Stirling's** formula as an example

18. Difference Equations 177
Expressed directly in terms of y_k (unlike DE's)
You can solve any difference equation exactly, linear or non-linear. Need BC's at t=0.
Constant coefficients => same characteristic poly as in DE theory, same solutions roughly.
The Digamma Function $\psi(x)$ and why it shows up in certain sums: $\psi(x+1) - \psi(x) = 1/(x+1)$
How related to the Gamma function: $\psi(x) = d/dx(\ln \Gamma(x)) = \Gamma'(x)/\Gamma(x)$ = Psi function
Certain difference equation gives the Fibonacci numbers
A major application will come later in approximating differential equations.

19. Differential Equations 193

Most general first order DE is y'(x) = f(x,y), use **isoclines**
This chapter talks about how to do *numerical* solutions of differential equations
Euler method says to iterate it: $y_{k+1} = y_k + h\, y'_k = y_k + h\, f(x_k, y_k)$
Taylor method keeps more terms: $y(x+h) = y(x) + h\, f(x,y) + 1/2\, h^2\, f'(x,y)$
Runge-Kutta is same idea, but use nearby mesh points instead of f'(x,y)
Predictor-Corrector methods: you iterate a pair of equations, not just one
Milne Method, **Adams** predictor, etc.

20 Differential Problems of Higher Order 223

Any higher order ODE can be rewritten as a system of first-order equations! (prob 20.5)
You can solve any *system* of first-order equations by methods of Chapter 19
An example is y" = f(x,y,y'), these claims apply to linear and non-linear.
For general order, you need BC's *all at t=0*, called initial conditions. Perhaps several y_k.

21. Least Squares Polynomial Approximation (L_2 norm) 235

Unlike collocation polys, these fits generally "miss" all the points, but are better fits.
Formalism for doing a least squares **linear fit** to discrete data, a matrix problem.
Reinterpretation in terms of **Hilbert Space** and **projection** and **normal equations**.
The general Hilbert Space ideas are expressed on page 243, derived in Stakgold Vol 1.
Do a fit to some points, get a curve, then **differentiate** THAT.
Do a fit to **smooth data** samples that have noise on them.
For large N, warning that normal equations have matrix that is **Hilbert Matrix** like.
Avoid this by working in an **orthogonal basis**: **Discrete Legendre functions**!
Repeat everything above but in continuous x space. Now **regular Legendres**.
The **Chebyshev** polys apply to norm that has odd square root weight. Equal error ripple.
Economization of a polynomial using Chebyshevs.

22. Min-max polynomial approximation 267

Discrete problem of fitting set of points with min abs value error over points.
The **Chebyshev line** fits 3 points with alternating sign but otherwise equal errors.
Fit more than 3 points this way using the **exchange method**.
Continuous data version of min-max. The **Weierstrass Theorem** that polys can get close.
Same 3-point alternating idea in continuous realm for linear fit. 3+n for order n fit.
Nice to express powers in $T_n(x)$ **Chebyshev polys** and truncate, then get min-max fit!
There is an **exchange method** in the continuous world as well.

23. Approximation by Rational Functions 283

Generalize idea of poly fit to fit as **poly/poly** to account for poles.
Usually quadratic /quadratic is good enough.
Can write poly/poly in odd **continued fraction** form with **reciprocal differences** ρ.

24. Trigonometric Approximation 293

Discrete sample "Fourier" analysis with sin, cos basis functions.
You are fitting a set of sample points with a sum of trig basis functions with coeffs.
For **continuous** data, this really is called **Fourier Series**. (no Fourier Integral stuff).

Since trigs are orthogs, truncation of series always gives **best least squares fit.**
The **Lanczos Sigma Factor** amounts to inserting a box filter into your problem.

25. Nonlinear Algebra 310

Subject is finding the **roots** of an arbitrary equation f(x) = 0.
Solving a **system** of equations like f(x,y) = 0 and g(x,y)= 0 is also treated.
Recast f(x) = 0 into the form F(x) = x, then **iterate** as $x_n = F(x_{n-1})$, slow convergence.
Aitken's Process and **Stephenson's method** of doing Aitken every 4th time.
The incomparable **Newton-Raphson method** with quadratic convergence.
The **regula falsi method** (1800 BC), an interpolation method.
The **Bernoulli method** to find the dominant root, and using Deflation to continue.
The **Quotient Difference method**.
Sturm sequences, very odd stuff
For system like f(x,y) = 0 and g(x,y)= 0, can use modified Newton-Raphson
method of steepest descent is another way to solve f(x,y) = 0 and g(x,y)= 0.
Bairstow's method for finding pair of complex roots.

26. Linear Systems 334

Subjects are: (1) A**x** = **b**, solve for **x**: (2) A**x** = λ**x**, solve for λ's and **x**'s
Fundamental Theorem of Linear Algebra: solve Ax=b if A is non-singular.
Gauss-elimination to get lower triangular form, then back-substitution to solve.
Gauss-Jordan elimination to get diag(ones) form, then you are done.
Gauss-Seidel relaxation method and the dogs in the corridors problem.
Other relaxation methods, how to accelerate them (**over-relaxation**)
The issue of the **numerical stability of a matrix** A. Wilson's Matrix.
Inverting a matrix using the **simplex swapping** method.
Fast **evaluation of a determinant** using Gauss elimination.
The **eigenvalue problem** (A-λ)**x** = **0** and how to find the **characteristic polynomial**.
How to find the largest (dominant) eigenvalue by the **power method**.
Using the **Rayleigh Quotient** to improve eigenvalue estimate.
The **Reduction Method** to find all eigenvalues.
Jacobi's Method: use ∞ iterative 2x2 ortho xforms to bring matrix to diagonal form.
Given's Method: use finite # of 2x2 orthos to get matrix to tri-diagonal form.
The can compute determinant iteratively and thus find the characteristic poly and roots.
The **eigenvalues** of an upper or lower triangular matrix are the diagonal elements.
How to handle Ax=b systems with things are **complex numbers**.
Old theorems: orthog(unitary) diagonalizes a real-symmetric(self-adjoint) matrix.

27. Linear Programming 361

The **Simplex Method** (I have a huge separate document on this, see also Glicksman book)
The **Duality Theorem** relates two linear programming problems in a simple way.
Two-Person Games (with a payoff matrix) form a dual set of linear programming problems.
In 1928 before Simplex 1947, in the game world von Neumann proved the **Min-Max theorem** which is now seen as just a special case of the duality theorem.

28. Overdetermined System (R is called the **residual**) 375

Suppose you are solving A**x**=**b** with some noisy data. Solve A**x** = **b** + **R** instead.
Try to minimize R with your solution for **x**, using either least squares or min-max.
The least squares method has a scalar product, and we can fit it into the Hilbert Space model
The min-max does not have a scalar product. Only way is to solve with Simplex Method.

29. Boundary Value Problems (ODE's with boundary values) 382

In our earlier solution of differences equations, we had "initial" boundary conditions *only*.
Here we have more general boundary conditions to think about, at a and b for example.
How to relate most general 2nd order *linear* ODE to our earlier difference equation problems.
Ie, how to solve initial value BC problems and lincom them to solve a,b BC problems.
Linear difference equations are **recursion formulas**.
As such, they result in **band matrices**, tridiagonal if ODE was 2nd order.
The a and b boundary conditions appear at the **ends** of the band matrix.
Solve the matrix problem then by **Gauss-Jordan elimination**, to approx solve your ODE.
For non-linear ODE's, use the **garden-hose method** (piss at the wall and adjust as needed)
Can associate the solution of an ODE with a Feynman type **extremal problem**.
Write your action, max or min the thing, then **Euler Equation** is your ODE.
This is **variational calculus**, see M&M for more.
Dynamic programming is a way to do the variational calculus thing.
Solving the **Diffusion Equation on a 2D mesh**, x and t.
Separation of variables exists in difference equation world as well as continuous world.
Diffusion equation with a **moving boundary**. (same as heat conduction equation, parabolic)
Solving the **Laplace Equation on a 2D mesh**, x and y. Use Gauss-Seidel relaxation. (elliptical)
Solving the **Wave Equation on a 2D mesh**, I think I see a propagator idea (hyperbolic)

30. **Monte Carlo Methods** 401

an alternative to trying full PDF solutions to statistical problems.
generating **pseudo-random numbers**
the **neutron** problem simulation
center of mass of particles on **rim of wheel** simulation
dog in corridors simulation