Summary
Advanced Computational Science

Dino Wernli, Fabian Hahn

October 4, 2009

1 Interpolation

1.1 Lagrange Interpolation

Given $n+1$ points (x_i, f_i), this method passes a polynomial of degree n through all the points as an approximation for $f(x)$. The formulas are:

$$f(x) \approx \sum_{i=0}^{n} f_i \cdot L_i(x)$$
$$L_i(x) = \prod_{j,j\neq i} \frac{x - x_j}{x_i - x_j}$$

It is obvious that this interpolation formula can be written as:

$$f(x) \approx \sum_{i=0}^{n} f_i \cdot L_i(x)$$
$$L_i(x) = \frac{1}{c_i} \cdot \frac{F(x)}{x - x_i}$$
$$c_i = \prod_{j,j\neq i} (x_i - x_j); \quad F(x) = \prod_{j} (x - x_j)$$

1.2 Hermite Interpolation

This method is based on $2n$ data points. It takes n tuples of the form (x_i, f_i, f_i') and passes a polynomial $P(x)$ of degree $2n-1$ through those points with the added property that the derivatives of $P(x)$ are equal to the derivatives f_i'.

The needed formulas for such an interpolation are as follows, where $U_k(x)$ and $V_k(x)$ are polynomials of degree $2n-1$:

$$f(x) \approx P(x) = \sum_{k=1}^{n} f_k \cdot U_k(x) + \sum_{k=1}^{n} f_k' \cdot V_k(x)$$
$$U_k(x) = [1 - 2 \cdot L_k'(x_k) \cdot (x - x_k)] \cdot L_k^2(x)$$
$$V_k(x) = (x - x_k) \cdot L_k^2(x)$$

The Hermite polynomials U_k and V_k have the following properties (as direct consequence of how they are constructed):

$$U_k(x_j) = \delta_{jk}, \quad U_k'(x_j) = 0$$
$$V_k(x_j) = 0, \quad V_k'(x_j) = \delta jk$$

These properties guarantee that you can "create" a function by summing up U_k and V_k, since an added U_k polynomial will never affect the derivatives of $P(x)$ and an added V_k will never affect the evaluations of $P(x)$.

2 Numerical Integration

We are looking for the solution of the definite integral

$$I = \int_a^b f(x)\ dx$$

To compute the result numerically we will use a technique called quadrature where $n+1$ equidistant quadrature points $\{(x_0, f(x_0)), \ldots, (x_n, f(x_n))\}$ are given.
The following notations will be used in the sections to come:

$$\Delta = x_{i+1} - x_i = \frac{b-a}{n}$$
$$x_{i+\frac{1}{2}} = x_i + \frac{1}{2} \cdot \Delta$$
$$I_i = \int_{x_i}^{x_i+1} f(x)\ dx$$

2.1 Rectangle Rule

This technique consists of taking the quadrature points and splitting up the interval $[a, b]$ in n subintervals $[x_i, x_{i+1}]$.
The integral of $f(x)$ on the subinterval $[x_i, x_{i+1}]$ is then approximated as follows:

$$I_i = \int_{x_i}^{x_{i+1}} f(x)\ dx \approx f(x_{i+\frac{1}{2}}) \cdot \Delta = I_{R_i}$$

Geometrically, the integral is approximated by the surface area of the rectangle with height $f(x_{i+\frac{1}{2}})$.
The numeric approximation of the original problem becomes:

$$\int_a^b f(x)\ dx = \sum_{i=0}^{n-1} I_i \approx \sum_{i=0}^{n-1} I_{R_i} = I_R$$

2.2 Trapezoid Rule

Like for the Rectangle Rule, $[a, b]$ is again split up in the n subintervals. Again, the integral of a subinterval $[x_i, x_{i+1}]$ is approximated, this time by:

$$I_i = \int_{x_i}^{x_{i+1}} f(x)\ dx \approx \frac{f(x_i) + f(x_{i+1})}{2} \cdot (x_{i+1} - x_i) = I_{T_i}$$

This corresponds to using the endpoints x_i and x_{i+1} as corners of a trapezoid and computing the surface area of the trapezoid analytically.
The numeric approximation of the original problem becomes:

$$\int_a^b f(x)\ dx = \sum_{i=0}^{n-1} I_i \approx \sum_{i=0}^{n-1} I_{T_i} = I_T$$

2.3 Simpson Rule

This rule is based on the observation that the Δ^3 summand in the subinterval error can be eliminated (more on this in the error section) by using the following linear combination of Rectangle and Trapezoid Rules:

$$I_i = \int_{x_i}^{x_{i+1}} f(x)\ dx \approx \frac{2I_{R_i} + I_{T_i}}{3} = I_{S_i}$$

The numeric approximation of the original problem becomes:

$$\int_a^b f(x)\ dx = \sum_{i=0}^{n-1} I_i \approx \sum_{i=0}^{n-1} I_{S_i} = I_S$$

Note that since the Simpson rule uses points on both ends of each interval and in the middle of each interval, it effectively needs more points.

2.4 Newton-Cotes Formula

Instead of splitting $[a, b]$, the idea here is to interpolate all the $n + 1$ points with a single polynomial $P(x)$ of degree n using the Lagrange interpolation method. You then obtain the approximation:

$$\begin{aligned}\int_a^b f(x)\ dx \approx \int_a^b P(x)\ dx &= \int_a^b \sum_{i=0}^{n} f(x_i) L_i(x)\ dx \\ &= \sum_{i=0}^{n} f(x_i) \int_a^b L_i(x)\ dx\end{aligned}$$

This gives us the Newton-Cotes formula:

$$\int_a^b f(x)\ dx \approx (b-a) \sum_{i=0}^{n} f(x_i) \cdot C_i^n$$

$$C_i^n = \frac{1}{b-a} \int_a^b L_i(x)\ dx$$

Note that Newton-Cotes corresponds to a global Trapezoid Rule for $n = 1$ and a global Simpson Rule for $n = 2$.

The Cotes-Numbers C_i^n have the following properties:

- Sum property: $\sum_{i=0}^{n} C_i^n = 1$
- Symmetry property: $C_i^n = C_{n-i}^n$

2.5 Error Estimates

Rectangle Rule Error

First, we take the taylor approximation of $f(x)$ around $x_{i+\frac{1}{2}}$ and integrate it in the interval $[x_i, x_{i+1}]$ to obtain the error:

$$\begin{aligned}\varepsilon_i =& I_{R_i} - I_i = I_{R_i} - \int_{x_i}^{x_{i+1}} f(x)\ dx \\ =& I_{R_i} - \int_{x_i}^{x_{i+1}} f(x_{i+\frac{1}{2}}) + f'(x_{i+\frac{1}{2}}) \cdot (x - x_{i+\frac{1}{2}}) + \dots\ dx \\ =& I_{R_i} - \underbrace{\int_{x_i}^{x_{i+1}} f(x_{i+\frac{1}{2}})\ dx}_{=f(x_{i+\frac{1}{2}}) \cdot \Delta = I_{R_i}} - \int_{x_i}^{x_{i+1}} f'(x_{i+\frac{1}{2}}) \cdot (x - x_{i+\frac{1}{2}})\ dx \\ & - \int_{x_i}^{x_{i+1}} \frac{1}{2!} f''(x_{i+\frac{1}{2}}) \cdot (x - x_{i+\frac{1}{2}})^2\ dx \\ & - \int_{x_i}^{x_{i+1}} \frac{1}{3!} f^{(3)}(x_{i+\frac{1}{2}}) \cdot (x - x_{i+\frac{1}{2}})^3\ dx - \dots\end{aligned}$$

It is easy to observe that the following holds for odd k:

$$\begin{aligned}&\int_{x_i}^{x_{i+1}} f^{(k)}(x_{i+\frac{1}{2}}) \cdot (x - x_{i+\frac{1}{2}})^k\ dx \\ &= f^{(k)}(x_{i+\frac{1}{2}}) \cdot \underbrace{\int_{x_i}^{x_{i+1}} (x - x_{i+\frac{1}{2}})^k\ dx}_{=0} = 0\end{aligned}$$

Therefore, since:

$$\int_{x_i}^{x_{i+1}} \frac{1}{2!} f''(x_{i+\frac{1}{2}}) \cdot (x - x_{i+\frac{1}{2}})^2\ dx = \frac{\Delta^3}{24} f''(x_{i+\frac{1}{2}})$$

the error ε_i becomes:

$$\begin{aligned}\varepsilon_i &= I_{R_i} - I_{R_i} - 0 - \frac{\Delta^3}{24} f''(x_{i+\frac{1}{2}}) - 0 - O(\Delta^5) \\ &= -\frac{\Delta^3}{24} f''(x_{i+\frac{1}{2}}) - O(\Delta^5) \in O(\Delta^3)\end{aligned}$$

Therefore, the estimate of the total integration error can be computed as follows:

$$\begin{aligned}\varepsilon &= -I_R + I = -I_R + \sum_{i=0}^{n-1} I_i = -I_R + \sum_{i=0}^{n-1} (I_{R_i} + \varepsilon_i) \\ &= -I_R + I_R - \sum_{i=0}^{n-1} \varepsilon_i = \sum_{i=0}^{n-1} \varepsilon_i \\ &\approx \sum_{i=0}^{n-1} -\frac{\Delta^3}{24} f''(x_{i+\frac{1}{2}}) = -\frac{\Delta^3}{24} \cdot \sum_{i=0}^{n-1} f''(x_{i+\frac{1}{2}}) \\ &\approx -\frac{\Delta^3}{24} \cdot n \cdot \langle f''(x_i) \rangle \\ &= -\frac{\Delta^2}{24} (b-a) \cdot \langle f''(x_i) \rangle \in O(\Delta^2)\end{aligned}$$

Trapezoid Rule Error

Identically to the Rectangle Rule, the 0-th summand and all summands of odd order in the Taylor Approximation of ε_i vanish. This leaves us with:

$$\varepsilon_i = \frac{\Delta^3}{12} \cdot f''(x_{i+\frac{1}{2}}) + O(\Delta^5)$$

Therefore, the total integration error ε is also in $O(\Delta^2)$.

Simpson Rule Error

The Simpson Rule is useful because the integration error of the subinterval $[x_i, x_{i+1}]$ is of order Δ^5. It is computed as follows:

$$\begin{aligned}\varepsilon_i &= I_i - I_{S_i} = I_i - \frac{2I_{R_i} + I_{T_i}}{3} \\ &= I_i - \frac{2I_i + 2\varepsilon_{R,i} + I_i + \varepsilon_{T,i}}{3} \\ &= \frac{1}{3}(2\varepsilon_{R,i} + \varepsilon_{T,i}) \\ &= \frac{1}{3}(-2 \cdot \frac{\Delta^3}{24} f''(x_{i+\frac{1}{2}}) + \frac{\Delta^3}{12} f''(x_{i+\frac{1}{2}})) + O(\Delta^5) \\ &= 0 + O(\Delta^5) \in O(\Delta^5)\end{aligned}$$

Applying the same reasoning as for the other rules, this would lead to a total error of order $O(\Delta^4)$ for the whole integration.

2.6 Richardson Extrapolation

Method Derivation

Assume we have an algorithm $k_0(h) \approx g$ which approximates the difficult (or impossible) to compute solution g of a problem.
Also assume that the approximation always gets better for a smaller h and the following property holds:

$$\lim_{h \to 0} k(h) = g$$

Then we can develop the approximation as a Taylor series around the actual solution g:

$$k_0(h) = g + c_1 h + c_2 h^2 + \dots$$

By definition, cutting h in half would make the approximation better:

$$k_0\left(\frac{h}{2}\right) = g + c_1 \frac{h}{2} + c_2 \frac{h^2}{4} + \dots$$

Taking a the following linear combination, we can gain an order of h in the error:

$$k_1(h) = 2k_0\left(\frac{h}{2}\right) - k_0(h) = g + c_2' \cdot h^2 + c_3' \cdot h^3 + \dots$$

In most applications, cutting h in half means doubling the cost of computation, so we gain a whole order in the error by just doubling the cost.
Iterating this procedure once more, you obtain:

$$k_2(h) = \frac{4 \cdot k_1\left(\frac{h}{2}\right) - k_1(h)}{4 - 1} = g + c''^3 h^3$$

This leads to the general recursive formula:

$$k_n(h) = \frac{2^n \cdot k_{n-1}\left(\frac{h}{2}\right) - k_{n-1}(h)}{2^n - 1}$$

It is easy to observe that for any n the whole scheme can be written as a combination of the original algorithm k_0 applied to many different h.

Error Analysis

First we note that the following holds:

$$k_0\left(\frac{h}{2}\right) - k_0(h) = \underbrace{-\frac{1}{2}c_1 h}_{} - \frac{3}{4}c_2 h^2 + \dots$$

$$\varepsilon = g - k_0\left(\frac{h}{2}\right) = \underbrace{-\frac{1}{2}c_1 h}_{} - \frac{1}{4}c_2 h^2 + \dots$$

Neglecting the terms of order 2 and above, we get the following approximation for the error of $k_0\left(\frac{h}{2}\right)$:

$$\varepsilon = g - k_0\left(\frac{h}{2}\right) \approx k_0\left(\frac{h}{2}\right) - k_0(h)$$

Romberg Integration

As an example, we inspect the total error while integrating a function using the Trapezoid Rule, where h is the size of an interval $[x_i, x_{i+1}]$. It can be proven that the following holds:

$$k_0(h) = g + c_2 h^2 + c_4 h^4 + \dots$$

We now look at the first step of the improvement:

$$k_0\left(\frac{h}{2}\right) = g + \frac{1}{4}c_2 h^2 + \frac{1}{16}c_4 h^4 + \dots$$

Since the $O(h)$ is already non-existant, we apply the scheme with a double step and eliminate the $O(h^2)$ term:

$$k_1(h) = \frac{4 \cdot k_0\left(\frac{h}{2}\right) - k_0(h)}{4 - 1} = g + c_4' h^4 + \dots$$

$$k_2(h) = \frac{4^2 \cdot k_1\left(\frac{h}{2}\right) - k_1(h)}{4^2 - 1} = g + c_6' h^6 + \dots$$

$$k_i(h) = \frac{4^i \cdot k_{i-1}\left(\frac{h}{2}\right) - k_{i-1}(h)}{4^i - 1}$$

2.7 Adaptive Quadrature

Motivation

Assume we have a function which we need to integrate on the interval $[a, b]$. Also assume that this function only changes a lot on a rather small interval within $[a, b]$.
Applying the Romberg Integration scheme on the whole integration to reduce the error would be very wasteful, since for most of the interval a low-level k_i is sufficiently precise.
On the interval where the function changes a lot, however, we want to augment the density of points and the precision of our algorithm k to take into account the changes of the function.

Algorithm

Concretely, the algorithm is applied as follows:

1. Break interval in smaller substeps.
2. Use RE to get an error estimate for $g - k_0(\frac{h}{2})$.
3. For each interval check if the error is small enough. If not, subdivide.

2.8 Gauss Quadrature

Until now, the integration methods we have seen were able to compute the exact integral of a polynomial of degree $n-1$ by using n arbitrary points (x_i, f_i) of the polynomial.
This method provides a guarantee of the exact integral of any polynomial of degree $2n-1$ also using n points. The only difference is that the n points have to be chosen in a specific way.

To provide this precision, Gauss Quadrature allows $2n$ degrees of freedom. These degrees of freedom are the points x_i and the weights w_i. The integral itself ends up looking as follows:

$$I = \sum_{i=1}^{n} w_i f(x_i)$$

The art consists of choosing the points and the weights correctly. The first step is the observation that any integral can be transformed as follows:

$$\int_a^b f(x)\, dx = \frac{b-a}{2} \int_{-1}^{1} f(x)\, d\xi, \quad \xi = \frac{2x - (a+b)}{b-a}$$

where $f(x)$ is taken at the point: $x = \dfrac{\xi \cdot (b-a) + (a+b)}{2}$

So finding clever points on an arbitrary interval $[a, b]$ can be reduced to finding clever points on the interval $[-1, 1]$. From now on, we therefore can generally assume that $a = -1$ and $b = 1$.

If $f(x)$ is a polynomial of degree $2n - 1$, it can be expressed exactly by its Hermite interpoland:

$$f(x) = \sum_{k=1}^{n} f(x_k) \cdot U_k(x) + \sum_{k=1}^{n} f'(x_k) \cdot V_k(x)$$

$$\int_{-1}^{1} f(x)\ dx = \sum_{k=1}^{n} f(x_k) \int_{-1}^{1} U_k(x)\ dx + \sum_{k=1}^{n} f'(x_k) \int_{-1}^{1} V_k(x)\ dx$$

If we now set

$$w_k = \int_{-1}^{1} U_k(x)\ dx, \quad v_k = \int_{-1}^{1} V_k(x)\ dx$$

$$\Rightarrow \quad I = \sum_{k=1}^{n} w_k f_k + \sum_{k=1}^{n} v_k f'_k$$

As we can see, if we somehow manage to choose the points x_k in a way that all the v_k vanish, we will be able to compute the exact integral of a function of degree $2n-1$ by only using the n points (x_k, f_k).

$$I = \sum_{k=1}^{n} w_k f_k$$

Choosing the x_k

We want to achieve the following for all $k = 1 \dots n-1$:

$$v_k = \int_{-1}^{1} (x - x_k) L_k^2(x)\ dx = \int_{-1}^{1} F(x) L_k(x)\ dx = 0$$

Note that if all n points are different and we can choose them freely, $F(x)$ is an arbitrary polynomial of degree n and the $L_k(x)$ are all of degree $n-1$.
If we now set $F(x)$ to be the n-th order Legendre Polynomial $P_n(x)$, we need to show that:

$$\forall 1 \le k \le n-1 : \int_{-1}^{1} P_n(x) L_k(x)\ dx = 0$$

If $P_i(x)$ is the i-th order Legendre Polynomial, it is of degree i and the following property holds for every n:

$$\forall 0 \le i < n : \int_{-1}^{1} P_n(x) P_i(x)\ dx = 0$$

Since the Legendre Polynomials of order $\le n-1$ form a complete set in the space of $(n-1)$-th order polynomials, every L_k can be written as:

$$L_k(x) = \sum_{i=0}^{n-1} c_i \cdot P_i(x)$$

Substituting the $L_k(x)$ in the above equation and applying the property of the Legendre Polynomials, we have successfully shown that indeed

$$\forall 1 \le k \le n-1 : v_k = \int_{-1}^{1} F(x) L_k(x)\ dx = 0$$

holds if we choose the x_k to be the roots of the n-th order Legendre Polynomial $P_n(x)$.

3 Numeric Differentiation

The problem we are looking at consists of, given discrete evaluations $y(x_i)$ of a function y, to approximate the derivative y' of the function at those discrete points.

The following methods assume that:

$$h := x_i - x_{i-1} = x_{i+1} - x_i$$

3.1 Interpolation with 2 Points

This method describes a way to approximate the derivative $y'(x_i)$ using y_{i-1} and y_{i+1}.
First, we take a look at the linear Lagrange interpolation of y on $[x_i, x_{i-1}]$:

$$y(x) \approx \frac{x - x_i}{x_{i-1} - x_i} y_{i-1} + \frac{x - x_{i-1}}{x_i - x_{i-1}} y_i$$

We now take the analytical first derivative of the approximation and insert the points x_i and x_{i-1} to get two different approximation possibilities for the derivative of y:

Forward Difference:

$$y'(x_{i-1}) \approx (D_+ y)(x_{i-1}) = \frac{y_i - y_{i-1}}{x_i - x_{i-1}}$$

or, from the view of point x_i:

$$y'(x_i) \approx (D_+ y)(x_i) = \frac{y_{i+1} - y_i}{x_{i+1} - x_i}$$

Backward Difference:

$$y'(x_i) \approx (D_- y)(x_i) = \frac{y_i - y_{i-1}}{x_i - x_{i-1}}$$

Central Difference:

Obviously, with the points $(x_{i-1}, y_{i-1}), (x_i, y_i), (x_{i+1}, y_{i+1})$ you can compute $(D_- y)(x_i)$ and $(D_+ y)(x_i)$.
Our final approximation consists of taking the average between the two approximations, thus resulting in:

$$y'(x_i) \approx \frac{1}{2} \cdot (D_- + D_+) y(x_i) = \frac{y_{i+1} - y_{i-1}}{x_{i+1} - x_{i-1}} = \frac{y_{i+1} - y_{i-1}}{2h}$$

Intuitively, this corresponds to the slope of the line between y_{i-1} and y_{i+1}.

3.2 Interpolation with 3 Points

As we did in the last section with 2 points, we can construct the Lagrange interpolation of $y(x)$ on the interval $[x_{i-1}, x_{i+1}]$ using 3 points:

$$\begin{aligned} y(x) \approx & \frac{(x - x_i)(x - x_{i+1})}{(x_{i-1} - x_i)(x_{i-1} - x_{i+1})} \cdot y_{i-1} \\ & + \frac{(x - x_{i-1})(x - x_{i+1})}{(x_i - x_{i-1})(x_i - x_{i+1})} \cdot y_i \\ & + \frac{(x - x_{i-1})(x - x_i)}{(x_{i+1} - x_{i-1})(x_{i+1} - x_i)} \cdot y_{i+1} \end{aligned}$$

Again, we derive this approximation with respect to x, assume equidistant points and insert the points x_{i-1}, x_i, x_{i+1} to obtain:

Forward Difference:

If we insert x_{i-1} into the derived interpoland:

$$y'(x_{i-1}) \approx \frac{-3y_{i-1} + 4y_i - y_{i+1}}{2h}$$

Backward Difference:

If we insert x_{i+1} into the derived interpoland:

$$y'(x_{i+1}) \approx \frac{y_{i-1} - 4y_i + 3y_{i+1}}{2h}$$

Approximation for $y'(x_i)$:

If we insert x_i into the derived interpoland:

$$y'(x_i) = \frac{y_{i+1} - y_{i-1}}{2h}$$

As we can see, we obtain the same formula as before, which was to be expected, since we are using the same points to approximate the derivative.

3.3 Taylor Approximation

The next idea is to observe the Taylor approximation of y around the point x_i:

$$y(x_{i+1}) = y_i + h \cdot y_i' + \frac{h^2}{2} \cdot y_i'' + O(h^3)$$
$$y(x_{i-1}) = y_i - h \cdot y_i' + \frac{h^2}{2} \cdot y_i'' + O(h^3)$$

Subtracting the two equations gives us:

$$y_{i+1} - y_{i-1} = 2h \cdot y_i' + O(h^3)$$
$$\frac{y_{i+1} - y_{i-1}}{2h} = y_i' + O(h^2)$$
$$\Rightarrow y_i' = \frac{y_{i+1} - y_{i-1}}{2h} + O(h^2)$$

As we can see, this results in the same approximation as for the interpolation method.

This approach can be generalized by taking a linear combination of the Taylor approximations:

$$a \cdot y(x_i - h) + b \cdot y(x_i) + c \cdot y(x_i + h)$$
$$= (a + b + c)y(x_i) + (c - a)hy'(x_i) + \frac{c + a}{2}h^2 \cdot y''(x_i) + O(h^3)$$

Using this linear combination, we can very easily get a formula for $y'(x_i)$, we just need to find coefficients a, b, c so that the following holds:

$$a + b + c = 0$$
$$(c - a)h = 1$$
$$\frac{c + a}{2} = 0$$

Solving this system results in $a = c = \frac{1}{2h}; b = 0$, which corresponds to the formula we obtained.

This approach can be used to get higher order derivatives too, but we will have to make more terms disappear at the beginning. This implies a greater number of coefficents, so we will also have to use more Taylor approximations, like $y(x_i + 2h), y(x_i - 2h)$.

4 Ordinary Differential Equations

4.1 Motivation Example

In 1916, Lanchester developed his equation of combat. It describes the relationship between the army sizes $R(t)$ and $B(t)$ of 2 opposing parties in a war.
Assuming that r and b describe the number of enemies which one own soldier kills (firing efficiency), Lanchester proposed the equations with initial values:

$$\frac{d}{dt}R = -b \cdot B, \quad \frac{d}{dt}B = -r \cdot R$$
$$R(t = 0) = R_0, \quad B(t = 0) = B_0$$

As an example application, one can devide the equations and get:

$$\frac{dR}{dB} = \frac{bB}{rR} \Rightarrow rR\ dR = bB\ dB$$
$$\int_{R(t)}^{R_0} rR\ dR = \int_{B(t)}^{B_0} bB\ dB$$
$$r \cdot ({R_0}^2 - R(t)^2) = b \cdot ({B_0}^2 - B(t)^2)$$

Now if we assume the system is in balance at time t:

$$R(t) = B(t) = 0 \quad \Rightarrow \quad b \cdot (B_0)^2 = r \cdot (R_0)^2$$

Using this result, one can conclude that if $R_0 = 3 \cdot B_0$ then $b = 9r$ has to hold for the system to be in equilibrium.

4.2 Reduction of ODEs

Nearly any high-order ODE

$$\frac{d^n y}{(dt)^n} = f(t, y, \ldots, y^{n-1})$$

can be reduced to an equivalent system of first-order ODEs by setting $y_1 = y', y_2 = y'', \ldots, y_j = y^{(j)}$. The system then becomes:

$$\frac{dy}{dt} = y_1$$
$$\frac{dy_{j-1}}{dt} = y_j, \quad j = 2 \ldots n - 1$$
$$\frac{dy_{n-1}}{dt} = f(t, y_1, \ldots, y_{n-1})$$

4.3 Types of ODEs

Boundary Value Problem

A general ODE of degree n has an inifinite number of solutions. More precisely, there are exactly n degrees of freedom in the solution function.
An ODE problem is called a boundary problem if, in addition to the ODE itself, boundary conditions are given in form of evaluations of the solution function and/or its derivatives.
Boundary value problems can have 1, 0, or an infinite number of solutions.

Initial Value Problem

Given an ODE of degree n and n initial values, the solution function is unique if each initial value concerns a different

derivative of this function. A well-posed initial value problem looks as follows:

$$\begin{aligned} y^{(n)}(t) &= f(t, y, y', \ldots, y^{(n-1)}) \\ y(t_0) &= y_0 \\ y'(t_1) &= y_1 \\ &\vdots \\ y^{(n-1)}(t_{n-1}) &= y_{n-1} \end{aligned}$$

In many cases all the t_i are chosen to be zero, but this is not a necessity.

4.4 Numeric Solution for ODEs

Since every higher-order ODE can be transformed into a system of first-order ODEs, this discussion will only concern first-order ODEs.
Given the initial value problem

$$\frac{d}{dt}y(t) = f(y,t), \quad y(t=0) = y_0$$

The general solution is given by:

$$\begin{aligned} y(t_n) &= y_0 + \int_0^{t_n} f(y,t)\ dt \\ y_{n+1} &= y_n + \int_{t_n}^{t_{n+1}} f(y,t)\ dt \end{aligned}$$

Since we want to solve this equation numerically, we will develop methods for getting an arbitrary amount of $(t, y(t))$ tuples, thus characterizing the function $y(t)$ completely.

Corner Point Rectangle Rule - Euler Methods

The explicit Euler method is given by applying the rectangle quadrature rule to the integral form of the solution formula:

$$\begin{aligned} y_{n+1} &= y_n + \int_{t_n}^{t_{n+1}} f(y,t)\ dt \\ y_{n+1} &= y_n + h \cdot f_n, \quad h = t_{n+1} - t_n \end{aligned}$$

It demonstrates a simple way to get from point y_n to point y_{n+1}. A similar alternative to explicit Euler is something called implicit Euler. The only difference is that you take f_{n+1} instead of f_n for the rectangle rule:

$$y_{n+1} = y_n + h \cdot f_{n+1}, \quad h = t_{n+1} - t_n$$

Since y_{n+1} is computed in terms of itself, you generally get a non-linear equation at this point. This is why the method is called implicit.

Middle Point Rectangle Rule - Leap Frog Method

A direct application of the middle point rule to the integral form of the solution yields:

$$y_{n+1} = y_n + h \cdot f_{n+\frac{1}{2}}$$

Since we don't have $y_{n+\frac{1}{2}}$, this is also an implicit method.
To work around this fact, we propose to use a time step of $2h$. You then obtain a strategy for getting the value of y_{n+1} if you know the values y_n and y_{n-1}.

The method is built on the middle point rule and it requires the given points to be equidistant:

$$\begin{aligned} y(t_{n+1}) &= y_{n-1} + \int_{t_{n-1}}^{t_{n+1}} f(y,t)\ dt \\ y_{n+1} &= y_{n-1} + 2h \cdot f_n \end{aligned}$$

Since only the initial value is given, we would first have to compute the first iteration using a one-point method like Euler, then proceed as described above.
Note that this is a second order method (as opposed to the first order Euler method).

Trapezoid Rule

Using the exact same approach, but applying the Trapezoid rule, you obtain the formula:

$$y_{n+1} = y_n + h \cdot \frac{f_n + f_{n+1}}{2}$$

Note that this is also an implicit method, since y_{n+1} is required.

Simpson Rule

Applying the same approach with Simpson's quadrature rule yields:

$$y_{n+1} = y_{n-1} + \frac{h}{3}\left(f_{n-1} + 4f_n + f_{n+1}\right)$$

Not only is this method implicit, it is also unstable.

Linearization of Implicit Methods

As a demonstration example, we take the implicit Trapezoid method for the solution of our ODE:

$$y_{n+1} = y_n + \frac{h}{2} \cdot (f_{n+1} + f_n) + O(h^3)$$

Since this method is a 2^{nd} degree method (3^{rd} degree error), it would be enough to insert an approximation for f_{n+1} which doesn't cause a greater order total error than h^3.
Therefore, the next step would be a Taylor expansion of $f(y,t)$ around y_n, where t is left constant at t_{n+1}:

$$\begin{aligned} f(y_{n+1}, t_{n+1}) = f(y_n, t_{n+1}) &+ \underbrace{(y_{n+1} - y_n)}_{\in O(h)} \cdot f_y(y_n, t_{n+1}) \\ &+ \underbrace{\frac{1}{2} \cdot (y_{n+1} - y_n)^2}_{\in O(h^2)} \cdot f_{yy}(y_n, t_{n+1}) \\ &+ O(h^3) \end{aligned}$$

Since f_{n+1} is multiplied with h in the trapezoidal formula, we only need to plug in the approximation up to the linear term, since in that case the error remains in $O(h^3)$. After doing so, you obtain the new method:

$$y_{n+1} = y_n + \frac{h}{2} \cdot \frac{f(y_n, t_{n+1}) + f(y_n, t_n)}{1 - \frac{h}{2} \cdot f_y(y_n, t_{n+1})}$$

This method has the advantage that it maintains stability and is still second order accurate, even though it does not require solving non-linear equations exactly.

Runge-Kutta Methods

To get from y_n to y_{n+1}, the idea here is to consider intermediate points and to take a linear combination of the intermediate points to get an estimate of y_{n+1} of higher accuracy.

$$\begin{aligned} y_{n+1} &= y_n + \gamma_1 \cdot k_1 + \gamma_2 \cdot k_2 \\ k_1 &= h \cdot f(y_n, t_n) \\ k_2 &= h \cdot f(y_n + \beta \cdot k_1, t_n + \alpha \cdot h) \end{aligned}$$

First we perform a Taylor expansion of k_2:

$$k_2 = h\left[f(y_n, t_n) + \beta k_1 \cdot f_y(y_n, t_n) + \alpha h \cdot f_t(y_n, t_n) + O(h^2)\right]$$

Putting it together and inserting the approximation for k_2, the method becomes:

$$\begin{aligned} y_{n+1} = y_n &+ (\gamma_1 + \gamma_2) h f(y_n, t_n) \\ &+ \gamma_2 \beta h^2 f(y_n, t_n) f_y(y_n, t_n) + \gamma_2 \alpha h^2 f_t(y_n, t_n) \end{aligned}$$

For our method to actually be correct, we need the parameters α, β, k_1, k_2.
To compute them, we look at the following Taylor expansion:

$$\begin{aligned} y_{n+1} &= y_n + h \cdot y_n' + \frac{h^2}{2} \cdot y_n'' + O(h^3) \\ &= y_n + h \cdot f_n + \frac{h^2}{2} \cdot (f(y_n, t_n) \cdot f_y(y_n, t_n) + 1 \cdot f_t(y_n, t_n)) \end{aligned}$$

Performing a simple coefficient comparison yields the system:

$$\begin{aligned} \gamma_1 + \gamma_2 &= 1 \\ 2\gamma_2 \beta &= 1 \\ 2\gamma_2 \alpha &= 1 \end{aligned}$$

Leaving α as a degree of freedom, you obtain:

$$\gamma_2 = \frac{1}{2\alpha}; \quad \beta = \alpha; \quad \gamma_1 = 1 - \frac{1}{2\alpha}$$

One special case for this method is $\alpha = \frac{1}{2}$, which yields the formula:

$$y_{n+1} = y_n + h \cdot f\left(y_n + \frac{h}{2} f(y_n, t_n), t_{n+\frac{h}{2}}\right)$$

Note that the Runge-Kutta method above has been derived for 2 steps and taylor expansions until the first order of h. Both these parameters can be varied at will.

For example, a 3-step Runge-Kutta method would look like:

$$\begin{aligned} y_{n+1} &= y_n + \gamma_1 \cdot k_1 + \gamma_2 \cdot k_2 \\ k_1 &= h \cdot f(y_n, t_n) \\ k_2 &= h \cdot f(y_n + \beta \cdot k_1, t_n + \alpha \cdot h) \\ k_3 &= h \cdot f(y_n + \eta \cdot k_1 + \xi \cdot k_2, t_n + \delta \cdot h) \end{aligned}$$

4.5 Stability of Numerical ODE Methods

Some algorithms, even though they are proven to have a certain accuracy, produce results which have little in common with the exact solution. This phenomenon is called instability and can have various reasons.
For example: if the exact result of a problem is bounded to a certain value, a stable algorithm will be bounded to that value aswell.

Generally, numerical methods can be divided in 3 classes:

- Stable: the solution does not grow unbounded for any choice of parameters.
  These methods are generally more expensive.
- Unstable: the solution grows unbounded for any choice of parameters.
  Here the accuracy is irrelevant.
- Partially stable: the solution remains bounded for certain parameter values.
  These methods demonstrate a tradeoff between accuracy and stability.

Our Model Problem

To analyze stability of numerical ODE methods, we would like to choose a suited model $f(x,t)$ for the problem:

$$\begin{aligned} \frac{dx}{dt} &= f(x,t) \\ &\approx f(x_0,t_0) + (t-t_0)\frac{\partial f}{\partial t}(x_0,t_0) + (x-x_0)\frac{\partial f}{\partial x}(x_0,t_0) \\ &= \alpha_0 + \alpha_1 \cdot t + \lambda x \end{aligned}$$

The general solution for this approximation of our problem is given as:

$$x(t) = \alpha_0 \cdot t + \frac{1}{2}\alpha_1 \cdot t^2 + e^{\lambda t}$$

As we can see, the main source of instability is the λ parameter, since it can cause exponential growth. Therefore, we will use the following simplified problem class as model problem for our analysis:

$$\frac{dx}{dt} = \lambda x, \quad \lambda = \lambda_R + i \cdot \lambda_I, \quad \lambda_R \leq 0$$

A few notes about the choice of this specific model problem:

- $\lambda_R \leq 0$ ensures that the solution will always be bounded
- The parameter λ_I is an oscillation factor. We will demand that stable algorithms reflect the oscillation of the exact result
- Since $\lambda \in \mathbb{C}$, many ODEs can be transformed in such a way that they fall into this category of problems

Transformation Example

As an example of how many problems can be transformed so that they fall into the description of our model problem, we look at:

$$\frac{d^2x}{(dt)^2} + \omega^2 x = 0$$

The first step is to make this a system of first order ODEs:

$$\begin{aligned} \frac{dx_1}{dt} &= x_2 \\ \frac{dx_2}{dt} &= -\omega^2 x_1 \end{aligned}$$

Or, in vector notation:

$$\frac{d}{dt}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = A \cdot \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}, \quad \boldsymbol{A} = \begin{pmatrix} 0 & 1 \\ -\omega^2 & 0 \end{pmatrix}$$

Our goal is now to get a diagonal matrix $\boldsymbol{A}$, so that the equations become completely separate. First, we take the EVD of $\boldsymbol{A}$:

$$\boldsymbol{A} = \boldsymbol{S}^{-1}\boldsymbol{\Lambda}\boldsymbol{S}, \quad \boldsymbol{\Lambda} = \begin{pmatrix} i\omega & 0 \\ 0 & -i\omega \end{pmatrix}$$

We the introduce a new variable:

$$\boldsymbol{z} = \begin{pmatrix} z_1 \\ z_2 \end{pmatrix} = \boldsymbol{S} \cdot \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \boldsymbol{S} \cdot \boldsymbol{x}$$

We plug in what we have so far and obtain:

$$\begin{aligned} \frac{d}{dt}\boldsymbol{x} &= \boldsymbol{A}\boldsymbol{x} = \boldsymbol{S}^{-1}\boldsymbol{\Lambda}\boldsymbol{S}\boldsymbol{x} \\ \frac{d}{dt}\boldsymbol{S}\boldsymbol{x} &= \boldsymbol{\Lambda}\boldsymbol{S}\boldsymbol{x} \\ \frac{d}{dt}\boldsymbol{z} &= \boldsymbol{\Lambda}\boldsymbol{z} \end{aligned}$$

As we can see, we now have 2 first order ODEs which both fit the description of our model problem.

General Principle

In general, a numerical method is bounded if it can be written as follows:

$$x_n = \sigma^n \cdot x_0, \quad |\sigma| \leq 1$$

The factor σ is called amplification factor. When considering systems of ODEs, σ can be an expression containing matrices. In such a case, the stability property must hold for all cases where you replace the matrices in σ by their Eigenvalues.

Stability of Explicit Euler

If we use explicit Euler to solve:

$$\frac{dx}{dt} = \lambda x, \quad x(0) = x_0$$

We get the following formula:

$$\begin{aligned} x_{n+1} &= x_n + \Delta t \cdot \lambda x_n = (1 + \lambda\Delta t) \cdot x_n \\ x_n &= (1 + \lambda\Delta t)^n x_0 \end{aligned}$$

We define the amplification factor σ so that:

$$x_n = \sigma^n x_0$$

In our case, this means that:

$$\sigma = 1 + \lambda\Delta t = (1 + \lambda_R\Delta t) + i \cdot (\lambda_I\Delta t)$$

Obviously, the numerical solution is bounded if $|\sigma| \leq 1$. So the stability region for explicit Euler is the circle of radius 1 around the point $(-1, 0)$ in the $\lambda_R\Delta t, \lambda_I\Delta t$ plane.
The stability region of explicit Euler only touches the imaginary axis at the point $(0, 0)$. This means that the algorithm will be unstable for purely complex λ.
We can also see that for purely real λ stability is granted for $|\lambda\Delta t| \leq 2$.

Stability of Implicit Euler

The same analysis for implicit Euler yields for $x_n = \sigma^n x_0$:

$$\begin{aligned} \sigma &= \frac{1}{1 - \lambda\Delta t} = \frac{1}{r \cdot e^{i\phi}} \\ r &= \sqrt{(1 - \lambda_R\Delta t)^2 + (\lambda_I\Delta t)^2} \\ \Rightarrow \quad |\sigma| &= \frac{|e^{-i\phi}|}{|r|} = \frac{1}{|r|} \leq 1, \text{ since: } \lambda_R \leq 0 \end{aligned}$$

This means that implicit Euler belongs to the unconditionally stable methods.

Stability of Two-Step Runge-Kutta

Applying the derived two-step formula to our model problem, we get:

$$\begin{aligned} x_{n+1} &= x_n + \left(1 - \frac{1}{2\alpha}\right)\lambda h \cdot x_n + \frac{1}{2\alpha}\lambda h(1 + \alpha\lambda h) \cdot x_n \\ &= x_n + \lambda h x_n + \frac{1}{2}\lambda^2 h^2 x_n \\ &= x_n \cdot \left(1 + \lambda h + \frac{1}{2}\lambda^2 h^2\right) \\ &\Rightarrow \sigma = 1 + \lambda h + \frac{1}{2}\lambda^2 h^2 \end{aligned}$$

For stability we therefore need:

$$\left|1 + \lambda h + \frac{1}{2}\lambda^2 h^2\right| < 1$$

In the complex plane, this corresponds to an elipse which completely contains the circular stability area of explicit Euler. A common point is that this method, like Euler, is unstable for purely imaginary λ.

4.6 Phase Errors of ODE Methods

When the exact solution of the ODE is purely oscillatory some numerical methods produce results which would be correct, but with a different phase. The difference in phase between the exact solution and the numerical one is called phase error.

For our model problem, a purely oscillatory solution means that $\lambda_R = 0$:

$$\frac{dx}{dt} = i\omega x \quad \Rightarrow \quad x(t) = e^{i\omega t}$$

Phase Error of Explicit Euler

Applying explicit Euler to this problem, we obtain:

$$\begin{aligned} x_{n+1} &= x_n + h \cdot i\omega \cdot x_n = x_n(1 + h \cdot i\omega) \\ x_n &= \sigma^n \cdot x_0, \quad \sigma = 1 + h \cdot i\omega \end{aligned}$$

Analyzing σ as a complex number, we obtain:

$$\begin{aligned} \sigma &= |\sigma| \cdot e^{i\phi} \\ |\sigma| &= \sqrt{1 + \omega^2 h^2}, \quad \phi = \tan^{-1}\left(\frac{\omega h}{1}\right) \end{aligned}$$

The actual phase error PE_n after the n-th step is given as:

$$\begin{aligned} PE_n &= \omega t_n - \phi_n = \omega(t_0 + nh) - (\phi_0 + n\phi) \\ &= \underbrace{\omega t_0}_{\phi_0} + \omega nh - \phi_0 - n\phi \\ &= n \cdot \left(\omega h - \tan^{-1}(\omega h)\right) \end{aligned}$$

Now consider the power series of the function $\tan^{-1}(x)$:

$$\tan^{-1}(x) = x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \ldots = x - O\left(x^3\right)$$

Substituting in the above equation and recalling that ω is a constant, we obtain:

$$PE_n \in O\left(nh^3\right)$$

Phase Error of Trapezoid Rule

In the same way as done above for Explocit Euler, it can be shown that even though the scheme is unconditionally stable in terms of boundedness ($|\sigma| = 1$), the phase error has the same order $O\left(nh^3\right)$ for the (implicit) Trapezoid Rule for the above problem.

5 Partial Differential Equations

5.1 Motivation Example

This discussion will be based on an example PDE which describes diffusion.
In our example, the notion of diffusion will be applied to the net migration of particles in air. The x-coordinate will be our only spacial coordinate and we will consider the spacial points x and $x + dx$.
First, we define $N(x)$ as the number particles in a "dx-large box around the point x".
We also define $J_x(x)$ as the net amount of particles flowing from the box around x to the box around $x + dx$ (per area and time unit). J_x is measured in $\frac{\#\text{ particles}}{\text{Time} \cdot \text{Area}}$.

The random flow of particles will generally cause a box to lose particles:

- to the left: a fraction f of the total particles
- to the right: the same fraction f of the total particles

Note that if we use this model, $f \leq \frac{1}{2}$ must hold.

We now consider the time step from a time t to the time $t + h$. The total number of particles flowing from box x to box $x + dt$ is $J_x(x) \cdot A \cdot h$ and can be computed as follows:

$$\begin{aligned} & J_x(x) \cdot A \cdot h = f \cdot N(x) - f \cdot N(x + dt) \\ \Rightarrow \quad & J_x(x) = -\frac{f \cdot (N(x + dx) - N(x))}{A \cdot h} \end{aligned}$$

We now multiply the right side of the equation with $\frac{dx^2}{dx^2}$ and obtain:

$$J_x = -f\frac{dx^2}{h} \cdot \frac{1}{dx}\left[\frac{N(x + dx)}{A \cdot dx} - \frac{N(x)}{A \cdot dx}\right]$$

We now define the concentration $C(x)$:

$$C(x) = \frac{N(x)}{A \cdot dx}$$

Note that since technically, $N(x)$ is a function of t, $C(x)$ must also be a function of t. Furthermore, since $N(x)$ is roughly linear in dx, the concentration $C(x)$ is independant of the spacial step-size dx (which makes sense).

Our formula thus becomes:

$$J_x = -\underbrace{f\frac{dx^2}{h}}_{=\text{ Diffusivity } D}[C(x + dx) - C(x)] \cdot \frac{1}{dx}$$

The last step is now to take the limit $dx \to 0$ and we obtain:

Fick's First Law

$$J_x(x) = -D\frac{\partial C(x,t)}{\partial x}$$

Fick's Second Law

This law states, that the overall number of particles must be conserved over time.
We now observe that during a time period h:,

- $J_x(x) \cdot A \cdot h$ particles enter the box x
- $J_x(x + dx) \cdot A \cdot h$ particles leave the box x

So, during a time step h, the number of particles added to box x can be expressed as:

$$J_x(x) \cdot A \cdot h - J_x(x + dx) \cdot A \cdot h$$

An alternative way to calculate the number of particles added to box x during h is:

$$N(x, t + h) - N(x, t) = A \cdot dx \cdot C(t + h) - A \cdot dx \cdot C(t)$$

Since both formulas describe the same notion, the following must hold:

$$\begin{aligned} & J_x(x) \cdot A \cdot h - J_x(x + dx) \cdot A \cdot h = A \cdot dx \cdot C(t + h) - A \cdot dx \cdot C(t) \\ & -\frac{J_x(x + dx) - J_x(x)}{dx} = \frac{C(t + h) - C(t)}{h} \end{aligned}$$

Taking the limits $h \to 0$ and $dx \to 0$ yields:

$$\begin{aligned} & \frac{\partial C}{\partial t} = -\frac{\partial J_x}{\partial x} \\ \Leftrightarrow \quad & \frac{\partial C}{\partial t} = -\frac{\partial}{\partial x}\left(-D\frac{\partial C}{\partial x}\right) \\ \Leftrightarrow \quad & \frac{\partial C}{\partial t} = D\frac{\partial^2 C}{\partial x^2} \end{aligned}$$

We will use this last equation as our final problem, which we will seek to solve numerically.

5.2 The PDE Problem

The full problem which we will want to solve is given as:

$$\frac{\partial C}{\partial t} = D\frac{\partial^2 C}{\partial x^2}, \quad t > 0, x \in [0, L]$$

We seek to find the function $C(x, t)$ which describes the concentration of particles at a given point in time and space, thus fully characterizing the diffusion process.
Note that a positive D always implies a decaying solution, so

we will want a numerical method to also produce decaying results (stability).
In addition to the PDE, we need the initial value:

$$C(x,0) = g(x)$$

We also need the boundary values:

$$C(0,t) = a(t)$$
$$C(L,t) = b(t)$$

5.3 Solution Using Euler's Method

Our first step in solving the problem above consists in discretizing the spacial dimension x. We now consider the x-axis as a discrete series of points x_i, where:

$$x_i = i \cdot \Delta x, \quad i = 0 \dots N+1, \ \Delta x = \frac{L}{N}$$

Now we approximate the second derivative at a fixed time t by the numerical finite difference scheme:

$$\frac{\partial^2}{\partial x^2} C(x_i,t) \approx \frac{C(x_{i+1},t) - 2C(x_i,t) + C(x_{i-1},t)}{\Delta x^2}$$

We then obtain one approximated problem for each inner x_i where $i = 1 \dots N$:

$$\frac{\partial}{\partial t} C(x_i,t) = D \cdot \frac{C(x_{i+1},t) - 2C(x_i,t) + C(x_{i-1},t)}{\Delta x^2}$$

Using the boundary condtions for the border points x_0 and x_{N+1} we obtain:

$$\frac{\partial}{\partial t} C(x_1,t) = D \cdot \frac{a(t) - 2C(x_1,t) + C(x_2,t)}{\Delta x^2}$$
$$\frac{\partial}{\partial t} C(x_N,t) = D \cdot \frac{C(x_{N-1},t) - 2C(x_N,t) + b(t)}{\Delta x^2}$$

As we can see, this is a system of first-order ODEs, which can be written as follows:

$$\frac{d}{dt}\boldsymbol{C} = A \cdot \boldsymbol{C} + \begin{pmatrix} a(t) \\ 0 \\ \vdots \\ 0 \\ b(t) \end{pmatrix}$$

$$A = \frac{D}{\Delta x^2} \cdot \begin{pmatrix} -2 & 1 & 0 & \dots & 0 \\ 1 & -2 & 1 & \ddots & \vdots \\ 0 & 1 & -2 & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & 1 \\ 0 & \dots & 0 & 1 & -2 \end{pmatrix}$$

The next step is to discretize the t dimension. For the continuation of the derivation, the following notation is introduced:

$$C_i^j = C(x_i, t_j)$$

The explicit Euler scheme for a fixed $C(x_i,t)$ becomes:

$$C_i^{j+1} = C_i^j + \Delta t \cdot D \frac{C_{i+1}^j - 2C_i^j + C_{i-1}^j}{\Delta x^2}$$

The order of computation is to always compute all the $C(x_i,t_j)$ for a fixed j, and then to use these to compute the all the $C(x_i,t_{j+1})$. All the evaluations $C(x_i,0)$ are provided by the initial value.

The result is then a grid of function values $C(x_i,t_j)$ for all i,j which represent the solution function $C(x,t)$.

5.4 Stability of Euler and Central Differences

In this chapter we will use a variety of different techniques to analyze the stability of the application of Euler and central differences to the PDE problem described above.

Classical Stability Analysis

For the second discretization, we would like to choose a time step Δt which leads to a stable explicit Euler scheme.
The Eigenvalues of A are the relevant values for this stability analysis, they are all negative (which is good since this means that the solution will be decaying) and can be computed as:

$$\lambda_i = \frac{D}{\Delta x^2}\left(-2 + 2\cos\left(\frac{\pi i}{N}\right)\right); \quad i = 1, \dots, N-1$$

The Eigenvalue of A which yields the most critical condition (in this case the Eigenvalue with the largest magnitude) is:

$$\lambda_{max} = -4 \cdot \frac{D}{\Delta x^2}$$

Considering the stability circle of explicit Euler, you then obtain the stability condition:

$$\Delta t < \frac{\Delta x^2}{2D}$$

Another very important property of this matrix is that if $\lambda_1 \le \dots \le \lambda_N$, then:

$$\frac{\lambda_N}{\lambda_1} = \frac{4N^2}{\pi^2}$$

For a high N, A is very ill-conditioned. One says, A is stiff.

Von Neumann Stability Analysis

We would like to compute the stability of:

$$u_j^{n+1} = u_j^n + \frac{D \cdot \Delta t}{\Delta x^2}(u_{j+1}^n - 2u_j^n + u_{j-1}^n)$$

Neumann postulated the following assumption regarding the form of u:

$$u_j^n = \rho^n \cdot e^{ikx_j}$$

Plugging this into the equation, we obtain:

$$\rho^{n+1} \cdot e^{ikx_j} = \rho^n \cdot e^{ikx_j} + \frac{D \cdot \Delta t}{\Delta x^2}\rho^n \left(e^{ikx_{j+1}} + e^{ikx_{j-1}} - 2e^{ikx_j}\right)$$
$$\rho = 1 + \frac{D \cdot \Delta t}{\Delta x^2}\left(2\cos(k\Delta x) - 2\right)$$

Since ρ is obviously the amplification factor of the solution, stability must require:

$$|\rho| \le 1 \quad \Rightarrow \quad \left|1 + \frac{2D \cdot \Delta t}{\Delta x^2} \cdot (\cos(k\Delta x) - 1)\right| \le 1$$

The most critical situation for the above equation occurs when $\cos(k\Delta x) = -1$, so we obtain the condition:

$$\left|1 + \frac{2D \cdot \Delta t}{\Delta x^2} \cdot (-2)\right| \le 1$$

The first case where

$$1 + \frac{2D \cdot \Delta t}{\Delta x^2} \cdot (-2) \geq 0$$

yields the useless result $\Delta t \geq 0$. However the second case leads us to:

$$-1 + 4\frac{D \cdot \Delta t}{\Delta x^2} \leq 1$$

After transforming this, we get the familiar result:

$$\Delta t < \frac{\Delta x^2}{2D}$$

Modified Wave Number Analysis

We first consider our PDE problem:

$$\frac{\partial \phi}{\partial t} = \nu \frac{\partial^2 \phi}{\partial x^2}$$

We now take an ansatz regarding the form of the analytical result $\phi(x,t)$ which looks as follows:

$$\phi(x,t) = \psi(t) \cdot e^{ikx}$$

Plugging this into the PDE yields:

$$\begin{aligned} & \frac{d\psi(t) \cdot e^{ikx}}{dt} = \nu \cdot \psi(t) \cdot i^2 k^2 e^{ikx} \\ \Leftrightarrow \quad & \frac{d\psi(t)}{dt} = -\nu \cdot \psi(t) \cdot k^2 \end{aligned}$$

Now we discretize in space using central differences and get the system of ODEs:

$$\frac{d}{dt}\phi_j = \nu \frac{\phi_{j+1} - 2\phi_j + \phi_{j-1}}{\Delta x^2}$$

We now take the same ansatz for ϕ but now considering the discretization:

$$\phi(x_j,t) = \psi(t) \cdot e^{ikx_j}$$

Plugging this into the discretized problem (where we note that the discrete points must be evenly spaced) yields:

$$\begin{aligned} & \frac{d\psi(t) \cdot e^{ikx_j}}{dt} = \nu \cdot \left[\frac{\psi(t) \cdot (e^{ikx_{j-1}} - 2e^{ikx_j} + e^{ikx_{j+1}})}{\Delta x^2} \right] \\ \Leftrightarrow \quad & \frac{d\psi(t)}{dt} = \psi(t) \cdot \nu \cdot \underbrace{\left[\frac{(e^{-ik\Delta x} - 2 + e^{ik\Delta x})}{\Delta x^2} \right]}_{-k'^2} \\ \Leftrightarrow \quad & \frac{d\psi(t)}{dt} = \psi(t) \cdot \nu \cdot \underbrace{\left[\frac{2\cos(k\Delta x) - 2}{\Delta x^2} \right]}_{-k'^2} \end{aligned}$$

As we can see, we have established a connection between the analytically derived k and the k' which the discretization will produce, obviously depending on the choice of Δx:

$$-k'^2 = \frac{2\cos(k\Delta x) - 2}{\Delta x^2} \quad \Rightarrow \quad k'^2 = \frac{2}{\Delta x^2}[1 - \cos(k\Delta x)]$$

Until now, the derivation was completely independant of our method of time integration for $\psi(t)$. We now consider the explicit Euler method, where we recall our model problem:

$$\frac{d}{dt}y = \lambda y$$

In our case we have $\lambda = -\nu k^2$, and we recall that Euler was stable if:

$$\Delta t \leq \frac{2}{|\lambda|} = \frac{2}{|\nu k^2|}$$

For stability, we require that $k = k'$, which leads us to:

$$\Delta t \leq \frac{\Delta x^2}{|\nu \cdot (1 - \cos k\Delta x)|}$$

Worst-case analysis for the cosine yields the well-known result:

$$\Delta t \leq \frac{\Delta x^2}{2\nu}$$

5.5 Solution Using Implicit Methods

Trapezoidal Rule

Let's now assume that we have chosen a spacial discretization $R(t)$ for the second derivative:

$$\frac{\partial^2 u_j}{\partial x^2} \approx R_j(t)$$

Our diffusion PDE then becomes:

$$\frac{\partial u_j}{\partial t} = \nu \cdot R_j(t)$$

Directly applying the trapezoidal rule to discretize time (instead of Euler) we get:

$$u_j^{n+1} - u_j^n = \nu \cdot \Delta t \cdot \frac{1}{2}\left(R_j^{n+1} + R_j^n\right)$$

Replacing $R_j(t)$ by the finite difference approximation directly yields the following equation:

$$\begin{aligned} & -\beta u_{j+1}^{n+1} + (1+2\beta)u_j^{n+1} - \beta u_{j-1}^{n+1} \\ & = \beta u_{j+1}^n + (1-2\beta)u_j^n + \beta u_{j-1}^n \\ & \text{where } \beta = \frac{\nu \cdot \Delta t}{2\Delta x^2} \end{aligned}$$

Considering the space discretization, $\boldsymbol{u}$ can be written as a vector of discrete points. This means that at every time step n the whole system of equations above has to be solved.
Even though Gauss is quadratic in the number of u_j, this system is tridiagonal and therefore only requires linear time.

Stability of Trapezoidal Method

The first step is to consider the transformation of our PDE into a system of ODEs for which we used the finite differences. From the modified wavenumber analysis we know that if we use the ansatz

$$u_j = \psi(t) \cdot e^{ikx_j}$$

and plug it into the spacially discretized PDE, we are left with the task of solving the ODE system:

$$\begin{aligned} & \frac{d\psi}{dt} = -\nu k'^2 \psi \\ & \text{where } k'^2 = \frac{2}{\Delta x^2}\left(1 - \cos\left(k \cdot \Delta x\right)\right) \end{aligned}$$

The discretization now reduced the PDE to the simpler problem of solving an ODE:

$$\frac{dy}{dt} = \lambda \cdot y$$
$$\text{where } \lambda = -\nu \cdot \frac{2}{\Delta x^2}\left(1 - \cos\left(k \cdot \Delta x\right)\right)$$

Now performing von Neumann stability analysis for the solution of the general ODE $\dot{y} = \lambda \cdot y$ using the trapezoidal integration rule, we get the stability condition:

$$\rho = \frac{1 + \frac{1}{2}\lambda\Delta t}{1 - \frac{1}{2}\lambda\Delta t} \leq 1$$

Since we know the λ for our specific problem, we can plug it in and obtain:

$$\rho = \frac{1 - \frac{\nu\Delta t}{\Delta x^2}\left[1 - \cos(k\Delta x)\right]}{1 + \frac{\nu\Delta t}{\Delta x^2}\left[1 - \cos(k\Delta x)\right]} \leq 1$$

Since we know that $\nu > 0$ and that our steps in time and space are also positive, we can conclude that the stability condition will always hold.

We have therefore proved that the combination of trapezoid rule with central differences unconditionally produces a stable iteration scheme for the solution of the diffusion PDE.

Another interesting property of this specific stability condition is that for $\Delta t \to \infty$ we get $\rho \to -1$. So an oscillating iteration can be interpreted as an indicator for a too large time step (and hence inaccurate results).

Implicit Euler Rule

Inserting the implicit Euler method into our spacially discretized problem yields:

$$u_j^{n+1} - u_j^n = \Delta t \cdot \frac{\nu}{\Delta x^2}\left(u_{j+1}^{n+1} - 2u_j^{n+1} + u_{j-1}^{n+1}\right)$$

After a few algebraic transoformations, you obtain the following system of equations for $\boldsymbol{u}$:

$$-\gamma u_{j+1}^{n+1} + (1 + 2\gamma)u_j^{n+1} - \gamma u_{j-1}^{n+1} = \gamma u_j^n$$
$$\text{where } \gamma = \frac{\nu}{\Delta x^2}$$

Again, the system to be solved is a tridiagonal matrix, so the solution can be computed in linear time. Still, a whole such system has to be solved at each time step.

Stability of Implicit Euler

Since we are still using central differences to convert the PDE into a system of ODEs, we may still use:

$$\lambda = -\nu \cdot \frac{2}{\Delta x^2}\left(1 - \cos\left(k \cdot \Delta x\right)\right)$$

Applying the von Neumann stability analysis to implicit Euler for the general problem $\dot{y} = \lambda y$ yields the stability condition:

$$\rho = \frac{1}{1 - \lambda\Delta t}$$

Plugging in the λ, we get the condition:

$$\rho = \frac{1}{1 + \frac{2\nu\Delta t}{\Delta x^2}\left(1 - \cos(k \cdot \Delta x)\right)} \leq 1$$

Hence, the combination of central differences and implicit Euler also yields an unconditionally stable algorithm for solving the diffusion PDE.

Note that the same amount of work has to be done for implicit Euler and trapezoidal rule. But we know that the trapezoidal rule has higher order accuracy.

Again we notice the property that for $\Delta t \to \infty$ we see that $\rho \to 0$. So results which unexplainably iterate towards zero can be indicators for a too large time step and hence inaccuracy.

5.6 Multidimensional PDEs

Solution with Euler and Central Differences

We now consider PDEs of the form:

$$\frac{\partial u(x,y,t)}{\partial t} = \nu\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right)$$

Applying central differences to each second derivative separately and then discretizing time with explicit Euler, you obtain the iteration:

$$u_{l,j}^{n+1} = u_{l,j}^n + \nu \cdot \Delta t\left(\frac{u_{l+1,j}^n - 2u_{l,j}^n + u_{l-1,j}^n}{\Delta x^2} + \frac{u_{l,j-1}^n - 2u_{l,j}^n + u_{l,j+1}^n}{\Delta y^2}\right)$$

The only difference to the 1-D case is that you start with a whole plane. The iteration then gives you the next plane using the last one.

Stability of Explicit Euler

Again we try to reduce this problem to an ODE which corresponds to our model problem. So we start with a modified wave number stability analysis to find λ. Our new ansatz is the following:

$$u(t) = \psi(t) \cdot e^{ik_1 x + ik_2 y}$$

Plugging this into the spacially discretized PDE with central differences in both spacial axes, you obtain:

$$\frac{d\psi(t)}{dt} = -\nu(k'^2_1 + k'^2_2) \cdot \psi(t)$$
$$k'^2_1 = \frac{2}{\Delta x^2}\left(1 - \cos(\Delta x \cdot k_1)\right)$$
$$k'^2_2 = \frac{2}{\Delta y^2}\left(1 - \cos(\Delta y \cdot k_2)\right)$$

Applying the classical explicit Euler stability condition $\Delta t \cdot |\lambda| \leq 2$ and plugging in λ, we obtain:

$$\Delta t \leq \frac{2}{\nu\left(\frac{2}{\Delta x^2}\left(1 - \cos(\Delta x \cdot k_1)\right) + \frac{2}{\Delta y^2}\left(1 - \cos(\Delta y \cdot k_2)\right)\right)}$$

Considering the worst case for k_1 and k_2, we get:

$$\Delta t \leq \frac{1}{2\nu\left(\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2}\right)}$$

More specifically, if we happen to have $\Delta x = \Delta y = h$, the condition would simplify to:

$$\Delta t \leq \frac{h^2}{4\nu}$$

Furthermore, if you were to add a spcial z-dimension, it would look as follows for $\Delta x = \Delta y = \Delta z = h$:

$$\Delta t \leq \frac{h^2}{6\nu}$$