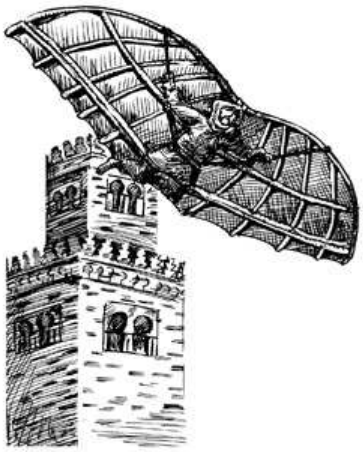

BODYandSOUL
MATHEMATICAL SIMULATION TECHNOLOGY
Draft

(To be downloaded and then opened to get links to work)

with contributions by Kenneth Eriksson, Don Estep, Peter Hansbo and Johan Hoffman

November 10, 2010

17
Newton's Laws of Motion

- 1st Law: In the absence of a net force, a body either is at rest or moves in a straight line with constant speed.
- 2nd Law: A body experiencing a force F experiences an acceleration a related to F by F = ma, where m is the mass of the body. Alternatively, force is equal to the time derivative of momentum.
- 3rd Law: Whenever a first body exerts a force F on a second body, the second body exerts a force -F on the first body. F and -F are equal in magnitude and opposite in direction.

17.1 Time-Stepping Newton's Equations of Motion

Newton's World is based on the following *incremental equations of motion* with smallest unit of time dt:

$$dx = vdt, \quad dv = adt, \tag{17.1}$$

as another way of writing

$$\frac{dx}{dt} = v, \quad \frac{dv}{dt} = a, \tag{17.2}$$

which combined with Newton's 2nd Law $F = a$ assuming $M = 1$, take the form:

$$dx = vdt, \quad dv = Fdt, \tag{17.3}$$

or

$$\frac{dx}{dt} = v, \quad \frac{dv}{dt} = F. \tag{17.4}$$

These equations are solved by time-stepping with time step dt:

$$dx^n = v^n dt \quad dv^n = a^n dt, \tag{17.5}$$

where

$$dx^n = x^{n+1} - x^n, \quad dv^n = v^{n+1} - v^n, \tag{17.6}$$

and $x^n = x(ndt)$ and $v^n = v(ndt)$ are the position and velocity at time ndt after n successive time steps with time step dt.

With each tick of time, velocity and position are thus updated according to

$$v^{n+1} = v^n + F^n dt, \quad x^{n+1} = x^n + v^n dt, \quad \text{for } n = 0, 1, ..., \tag{17.7}$$

from given initial values $v(0)$ and $x(0)$ at initial time $t = 0$, where $F^n = F(ndt)$ is the force acting on the body at time ndt. We refer to this update formula as *Euler's method* also called *Forward Euler*.

An alternative update formula is obtained by updating first velocity to v^{n+1} and using this value when updating to x^{n+1}:

$$v^{n+1} = v^n + F^n dt, \quad x^{n+1} = x^n + v^{n+1} dt, \tag{17.8}$$

which we will refer to as *Smart-Euler's method*. You will soon discover the difference between Euler and Smart-Euler.

A variant of Smart-Euler is

$$v^{n+1} = v^n + F^n dt, \quad x^{n+1} = x^n + \frac{1}{2}(v^n + v^{n+1})dt, \tag{17.9}$$

where the mean velocity $\frac{1}{2}(v^n + v^{n+1})$ is used instead of either v^n or v^{n+1}.

Below we shall meet variants with F^n depending on x^{n+1}. The basic method of this form is the *Trapezoidal Method*:

$$v^{n+1} = v^n + \frac{1}{2}(F^n + F^{n+1})dt, \quad x^{n+1} = x^n + \frac{1}{2}(v^n + v^{n+1})dt, \tag{17.10}$$

where $F^n = F(ndt, x^n)$ and $F^{n+1} = F((n+1)dt, x^{n+1})$, which requires iteration because F^{n+1} depends on x^{n+1}, which depends on v^{n+1}.

Below we shall recover Midpoint Euler in the form of the *continuous Galerkin cG(1)*, and *Backward Euler* with v^n and F^n in (17.7) replaced by v^{n+1} and F^{n+1}, as *discontinuous Galerkin dG(0)*.

We also refer to the Trapezoidal Method as *Midpoint Euler*, with Forward and Backward Euler as "Endpoint Euler".

We distinguish between *explicit methods* like Forward Euler with direct update, and *implicit* methods requiring *iteration*, like Midpoint Euler or Backward Euler, where the update formula for v^{n+1} and F^{n+1} is repeated with latest values inserted in the righthand side. With a (small) fixed number of iterations, implicit methods can be viewed as explicit direct update methods.

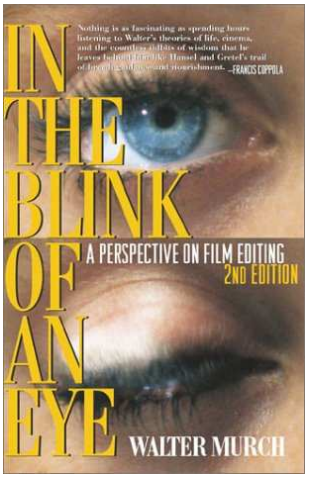

FIGURE 17.1. Eyeblink as time step.

17.2 Basic Solutions of the Equations of Motion

Newton's equations of motion, with given initial position x^0 and velocity v^0, take the form

$$\frac{dv}{dt} = \frac{F}{m}, \quad \frac{dx}{dt} = v \quad \text{for } t > 0, \quad x(0) = x^0,\ v(0) = v^0. \tag{17.11}$$

If $F = 0$, then the solution is given by

$$v(t) = v^0, \quad x = v^0 t + x^0 \quad \text{for } t \geq 0, \tag{17.12}$$

because if $v(t) = v^0$ then $dv = 0$, and if $x(t) = v^0 t$ then $dx = v^0 dt$.

If $F = 2$, $m = 1$ and $v^0 = 0$, then the solution is given by

$$v(t) = 2t, \quad x = t^2 + x^0 \quad \text{for } t \geq 0, \tag{17.13}$$

because if $v(t) = 2t$ then $dv = 2dt$, and if $x(t) = t^2$ then $dx = (t+dt)^2 - t^2 = (t + dt + t)\,dt \approx 2t\,dt$ using the formula

$$a^2 - b^2 = (a + b)(a - b). \tag{17.14}$$

By combination, we thus obtain the following solution formula for the basic case with F constant:

$$v(t) = \frac{F}{m}t + v^0, \quad x(t) = \frac{F}{m}\frac{t^2}{2} + v^0 t + x^0. \tag{17.15}$$

It is important that you understand the derivation of this formula. The key is to understand that $\frac{dv}{dt} = 1$ if $v = t$ and that $\frac{dx}{dt} = 2t$ if $x = t^2$.

17.3 The Fight: Newton vs Leibniz

Leibniz and Newton developed the basics of Calculus independently, in the second half of the 17th century. Newton accused Leibniz for plagiarism backed by the Royal Society of London, which made Leibniz very unhappy in his later years, see Newton vs. Leibniz. Of course, since Leibniz was such an honest scientist, he did not steal anything from Newton. In fact, it is Leibniz' Calculus which is now taught, which is a machine for symbolic and numerical computation with derivatives and integrals, and not Newton's theory of fluxions based on geometric arguments which is very difficult to understand and use.

17.4 Crash Test

- Crash test simulation
- Crash experiment

17.5 Watch

- Newton's Laws
- Newton's 2nd Law
- Conservation of Momentum.

17.6 Conservation of Momentum and Kinetic Energy

The *momentum* m of a body of mass M traveling with velocity v is defined by $m = Mv$. If the body is not acted upon by any force ($F = 0$), then *momentum is conserved*:

$$\frac{dm}{dt} = \frac{d}{dt}(Mv) = M\frac{dv}{dt} = Ma = F = 0 \tag{17.16}$$

If the body is acted upon by a force F, then momentum m changes according to

$$\frac{dm}{dt} = F \tag{17.17}$$

If we multiply this equation by v and interprete $Fv = W$ as *rate of work* W, then we we can write

$$\frac{dk}{dt} = \frac{d}{dt}\frac{Mv^2}{2} = Mv\frac{dv}{dt} = M\frac{dv}{dt}v = Fv = W, \tag{17.18}$$

where $k = \frac{Mv^2}{2}$ is the *kinetic energy*. We here used the fact that $\frac{d}{dt}v^2 = 2v\frac{dv}{dt}$, which we will prove shortly. We conclude that the kinetic energy changes according to

$$\frac{dk}{dt} = W = Fv. \tag{17.19}$$

In particular, if $F = 0$ then the kinetic energy is conserved.

For a system of particles interacting by elastic collisions, total momentum and kinetic energy are conserved if exterior forces vanish, because interior forces and work cancel.

If the velocity is a a vector $v = (v_1, v_2, v_3)$, so is momentum Mv, while kinetic energy K is a number (scalar)

$$k = \frac{M|v|^2}{2} = \frac{M(v_1^2 + v_2^2 + v_3^2)}{2}. \tag{17.20}$$

If we agree to generalize *conservation of momentum* to $\frac{dm}{dt} = F$ and *conservation of kinetic energy* to $\frac{dk}{dt} = W(= Fv)$, then we understand that

- Conservation of momentum is the same as Newton's 2nd Law.
- Conservation of kinetic energy is obtained by multiplying Newton's 2nd Law by velocity.

You will find these insights very helpful below.

17.7 Does Time-Stepping Respect Conservation of Kinetic Energy?

When you start to compute with Forward Euler, Smart Euler and Midpoint Euler, you will find that Forward Euler gains kinetic energy, Smart Euler loses kinetic energy, while Midpoint Euler as a compromise essentially conserves kinetic energy, in problems where kinetic energy should be conserved. You will also discover that the loss and gain decrease with decreasing time step.

We shall meet conservation of energy in a more general context in the next chapter, as conservation of *total energy* as the sum of kinetic energy and *potential/elastic energy*

17.8 To Think About

- How did Newton discover the 2nd Law?
- Who won the War of Calculus, Newton or Leibniz?

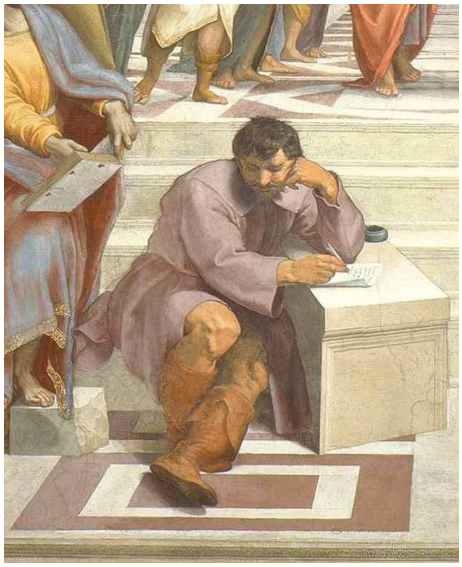

FIGURE 17.2. Heraclitus in Raphael's School of Athens: *Pantha rei...Everything flows...There is nothing permanent except change... .Much learning does not teach understanding...No man ever steps in the same river twice, for it's not the same river and he's not the same man...The eyes are more exact witnesses than the ears...Justice will overtake fabricators of lies and false witnesses...Big results require big ambitions...The way up and the way down are one and the same... Man is most nearly himself when he achieves the seriousness of a child at play...Men who wish to know about the world must learn about it in its particular details.* .

- Given the velocities of two elastic spheres about to impact, seek the velocities after impact. Conservation of (total) momentum? Conservation of (total) kinetic energy?

17.9 To Think About: Airbus 340-600

Consider the following data for an Airbus 340-600:

- take-off weight W: 368 tons
- wing area S: 439 square meter
- wing load $\frac{W}{S}$: 8383 Newton/square meter
- sea-level thrust T: 4×25.4 tons
- $\frac{W}{T} = 3.62$

- seats 380.

What is the take-off time and distance? What is the power required at crusing? Why is the engine power given as thrust in tons rather than horsepowers?

17.10 To Think About: Fokker 50

Consider the following data for a Fokker 50:

- take-off weight W: 20 tons
- wing area S: 70 square meter
- wing load $\frac{W}{S}$: 3000 Newton/square meter
- engine power P: 2×2050 kWatts
- $\frac{P}{W}$: 100 Watts/kilo
- crusing speed: 526 km/hour
- seats 50.

How many kW are required at crusing if $F = 10$ (which means that the thrust is 2 tons)? Are the engines oversized (for crusing?)?

17.11 To Watch: Airbus 340-600

- Crash 340-600 April 16 2009.
- Take off
- Emergence landing on Hudson River
- Construction in 116 seconds.

17.12 To Watch: Spitfire

- The story
- Spitfire vs MX2
- Start off

FIGURE 17.3. Airbus 340-600 and Supermarine Spitfires on mission.

- Under bridge

The Supermarine Spitfire is a British single-seat fighter aircraft used by the Royal Air Force and many other Allied countries through the Second World War. Specifications (Spitfire Mk Vb): max weight 3000 kg, engine Rolls-Royce Merlin 45 supercharged V12 engine, 1,470 hp at 9,250 ft (1,096 kW at 2,820 m), max speed 605 km/hour.

17.13 To Think About: Take-Off

To accellerate an airplane of weight W kp from rest to 60 meter/second in 60 seconds, requires an accelleration of 1 meter/second squared, that is a force of W Newton. For a jumbojet of 400 tons a thrust of 40 tons is required (because 1 kp is about 10 Newton), and the length of the starting lane is $\frac{1}{2}60^2 = 1.800$ meters. Doubling the thrust to 80 tons, reduces the time to 30 seconds and the starting lane to 900 meters, which is more realistic. To cruise at a finesse $F = 20$ requires a thrust of $\frac{400}{20} = 20$ tons, about a quarter of the thrust needed for take-off.

Can you figure out how much the length of the starting lane increases if you take into account that the drag increases as velocity squared, and thus the engine power available for acceleration decreases with speed (until the maximum speed is attained and no further accelleration is possible).

17.14 To Think About: Galileo's Experiment

Suppose you drop at the same time a tennis ball and a much heavier similar size pétanque (boule) ball from the Tower of Pisa, like Galileo did? How much quicker will the pétanque ball reach the ground? Compare the Reference Frame. Can you scale the balls so that they fall equally fast?

18
Particle-Spring System

Fear always springs from ignorance (Ralph Waldo Emerson, American Poet, Lecturer and Essayist, 1803-1882)

Let $x(t)$ be the position at time t of a unit point mass or *particle* moving without friction along a line subject to a linear spring force $F(x) = -x$. See Intro to Springs.

Newton's equations of motion take the form:

$$dx = vdt, \quad dv = -xdt. \tag{18.1}$$

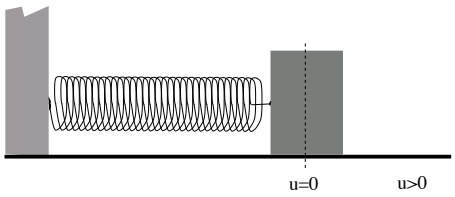

FIGURE 18.1. Particle-spring system: One particle/mass gliding without friction along a line attached to one of a spring attached to a fixed wall: Here $u(t) = x(t)$ is the position at time t measured from some the reference point with zero spring force

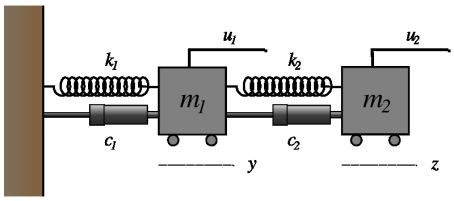

FIGURE 18.2. A 2particle-2spring system with dampers

18.1 Watch

- Particle-Spring
- Particle-Spring Frequency Response
- Particle-Spring Cows on Ice

18.2 To Think About

- How does a spring function?
- How to motivate that spring force is proportional to elongation?

18.3 Conservation of Total Energy

The total energy E a particle connected to a linear spring modeled by $\dot{x} = v$ and $\dot{v} = -x$ is defined by

$$E = \frac{1}{2}(x^2 + v^2). \tag{18.2}$$

Let us now prove that Midpoint Euler conserves the total energy. This follows by multiplying the time-stepping equations

$$x^{n+1} - x^n = \frac{1}{2}(v^{n+1} + v^n)dt, \quad v^{n+1} - v^n = -\frac{1}{2}(x^{n+1} + x^n)dt$$

by $\frac{1}{2}(x^{n+1} + x^n)$ and $\frac{1}{2}(v^{n+1} + v^n)$, respectively, to get by summation and reordering (using that $(a+b)(a-b) = a^2 - b^2$),

$$E^{n+1} \equiv \frac{1}{2}((x^{n+1})^2 + (v^{n+1})^2) = \frac{1}{2}((x^n)^2 + (v^n)^2) \equiv E^n \tag{18.3}$$

which expresses conservation of the total energy as $E^{n+1} = E^n$.

We understand that as the particle moves back and forth, kinetic energy is transformed into elastic energy stored as the spring stretches or compresses, which is transformed back into kinetic energy as the stretching and compression is eased.

19
Planetary System

> I demonstrate by means of philosophy that the earth is round, and is inhabited on all sides; that it is insignificantly small, and is borne through the stars. (Johannes Kepler)

The equations of motion for a planet (viewed as a pointlike particle) of unit mass orbiting a fixed Sun of unit mass centered at the origin, take the form

$$dx = vdt, \quad v = Fdt, \tag{19.1}$$

where

$$F(x) = -\frac{x}{|x|^3} \tag{19.2}$$

is the *gravitational force*. This is a force acting at distance, because the origin is the Sun at the origin, and it acts at x with distance $|x|$ from the origin.

Note that (19.2) is Newton's famous inverse square law of gravitation stating that the magnitude of the gravitational force F between two bodies with mass M_1 and $M-2$ at distance r is given by

$$F = G\frac{M_1 M_2}{r^2}, \tag{19.3}$$

where G is the gravitational constant.

We shall prove below that (19.2) this is a consequence of the fact that the gravitational potential satisfies a certain differential equation named Laplace's equation, and we shall uncover the assumptions leading to Laplace's equation. We can this way motivate that the exponent in Newton's Law is 2 and nothing else.

FIGURE 19.1. Jupiter.

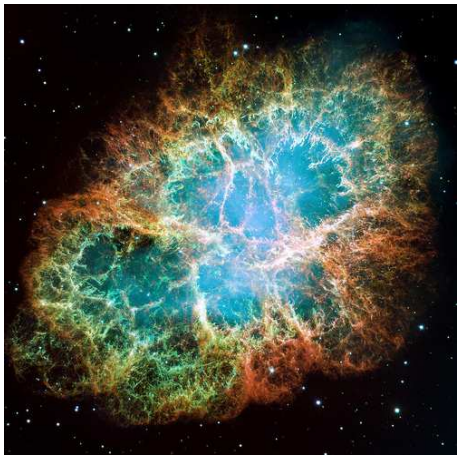

FIGURE 19.2. The Crab nebulosa: A macroscopic particle system.

FIGURE 19.3. Galileo presenting mathematical arguments to disbelieving Catholic priests.

FIGURE 19.4. Galileo's telescope.

19.1 Watch

- Poincaré and the chaos of the three-body problem.

19.2 To Think About

- What are Kepler's Laws?
- What is the simplest solution of a 2-body problem?

19.3 To Read

- BS How to prove Kepler's laws yourself.
- BS Solar System

19.4 Watch

- Kepler's Laws I
- Kepler's Laws II.

39
Particle-Spring Systems

I do not keep up with the details of particle physics. (Murray Gell-Mann)

It's indeed surprising that replacing the elementary particle with a string leads to such a big change in things. I'm tempted to say that it has to do with the fuzziness it introduces. (Edward Witten)

There are no Quantum Jumps, nor are there Particles! (H. D. Zeh)

39.1 Equations of Motion

We now generalize from pointlike hard particles to flexible systems consisting of hard particles connected by elastic springs, referred to as *particle-spring systems*. We start with a system consisting of two particles of mass M_1 and M_2, the positions of which we record by the coordinates $x^1(t)$ and $x^2(t)$. We connect the particles by an elastic spring with rest length L_{12} and spring constant E, which establishes a force between the particles, with the force acting on M_1 given by

$$F_{12} = E(r_{12} - L_{12})e_{12}, \tag{39.1}$$

where $r_{12} = |x^1 - x^2|$. This is an attractive force if $|r_{12}| < L_{12}$ and repulsive if $|r_{12}| > L_{12}$, and let

$$e_{12} = \frac{x^2 - x^1}{r_{12}} \tag{39.2}$$

be the vector of unit length pointing from x^1 to x^2. Of course (why?) the force F_{21} acting on particle M_2 is the reverse of F_{12} so that $F_{21} = -F_{12}$ (Newton's 3rd Law).

We say that this is a *linear spring* since the spring force is directly proportional to the elongation $|r_{12}| - L_{12}$ from the rest length.

The equations of motion are

$$\begin{aligned} \dot{x}^1(t) &= v^1(t), \quad \dot{x}^2(t) = v^2(t), \\ \dot{v}^1(t) &= \frac{F_{12}}{M_1}, \quad \dot{v}^2(t) = \frac{F_{21}}{M_2}, \end{aligned} \tag{39.3}$$

or in incremental form using Smart-Euler:

$$\begin{aligned} v^{1,n+1} &= v^{1,n} + \frac{F_{12}^n}{M_1}dt, \quad v^{2,n+1} = v^{2,n} + \frac{F_{21}^n}{M_2}, \\ x^{1,n+1} &= x^{1,n} + v^{1,n+1}dt, \quad x^{2,n+1} = x^{2,n} + v^{1,n+1}dt, \end{aligned} \tag{39.4}$$

where $x^{i,n} = x^i(ndt)$, $v^{i,n} = v^i(ndt)$ for $i = 1, 2$, and $F_{12}^n = E(r_{12}^n - L_{12})e_{12}^n$ with $e_{12}^n = \frac{x^{2,n} - x^{1,n}}{r_{12}^n}$ and $r_{12}^n = |x^{2,n} - x^{1,n}|$.

39.2 Experiments

- Flying Circus Cow
- Inside Flying Circus Cow
- Particle-spring elastic system

39.3 Generalization

We can directly generalize to any number of particles connected by any set of linear springs, including crossing springs. We can generlize to non-linear springs with a non-linear relation between spring force and spring elongation. See e.g. N-Body Systems.

39.4 Demo + Lab

- Test, Modify and Create Yourself (particlespring)

62
$x(t) = \int_0^t v(s)ds$ solves $\dot{x}(t) = v(t)$

> Without mathematics we cannot penetrate deeply into philosophy. Without philosophy we cannot penetrate deeply into mathematics. Without both we cannot penetrate deeply into anything. (Leibniz)

62.1 The Most Basic IVP

The solution of the IVP of finding $x : [0, T] \to \mathbb{R}$ such that

$$\dot{x}(t) = v(t) \quad \text{for } 0 < t \leq T, \quad x(0) = 0, \tag{62.1}$$

where $v : [0, T] \to \mathbb{R}$ is a given function and $[0, T]$ a given time-interval, is denoted by

$$x(t) = \int_0^t v(s)\, ds, \quad t \in [0, T], \tag{62.2}$$

and is referred to as the *integral* or *primitive function* of $v(t)$. So far the integral $\int_0^t v(s)\, ds$ is just a sign or name of the solution $x(t)$, and it remains to give it a concrete meaning. We shall see that the S-like integral sign $\int$ can be viewed as indicating a certain form of Summation, which we shall make precise. The integral sign $\int$ was the strike of genius of Leibniz, long before logotypes became the carriers of the inner meaning of companies and organizations.

The Forward Euler method for the IVP (62.1), is given by

$$x((n+1)dt) = x(ndt)+v(ndt)dt \quad \text{for } n = 0, 1, 2, ..., N, \quad \text{with } (N+1)dt = T, \tag{62.3}$$

or equally well

$$x((n+1)ds) = x(nds)+v(nds)ds \quad \text{for } n = 0, 1, 2, ..., N, \quad \text{with } (N+1)ds = T, \tag{62.4}$$

with $dt = ds$ the time step. If we replace $x(nds)$ by $x((n-1)ds) + v((n-1)ds)ds$, and so on, we see that $x((n+1)ds)$ can be expressed as as a sum

$$x((n+1)ds) = \sum_{m=0}^{n} v(mds)ds = v(0)ds+v(ds)ds+v(2ds)ds+...+v(nds)ds. \tag{62.5}$$

We are thus led to view

$$\int_0^t v(s)ds \quad \text{and} \quad \sum_{m=0}^{n} v(mds)\, ds, \tag{62.6}$$

to be similar, which we shall make precise below. We refer to the sum representation of the integral as a *Riemann sum*. We sum up so far:

Observation 1: The integral $x(t) = \int_0^t v(s)ds$ satisfies by definition

$$\dot{x}(t) = \frac{d}{dt}\int_0^t v(s)ds = v(t) \quad \text{for } 0 < t \leq T. \tag{62.7}$$

The integral $x(t) = \int_0^t v(s)ds$ represents a Riemann sum $\sum_{m=0}^{n} v(mds)\, ds$ with $(n+1)dt = t$.

Observation 2: The solution of the IVP, with possibly non-zero initial value x^0, of finding $x : [0, T] \to \mathbb{R}$ such that

$$\dot{x}(t) = v(t) \quad \text{for } 0 < t \leq T, \quad x(0) = x^0, \tag{62.8}$$

is given by

$$x(t) = x^0 + \int_0^t v(s)ds. \tag{62.9}$$

This is because the derivative of a constant function (the function $w(t) = x^0$), is zero ($\dot{w} = 0$).

Observation 3: Since $\dot{x}(t) = v(t)$ and $\dot{v}(t) = a(t)$ with $x(t)$ distance, $v(t)$ velocity and $a(t)$ accelleration, we can say that

- *distance is the integral of velocity*,
- *velocity is the integral of accelleration*.

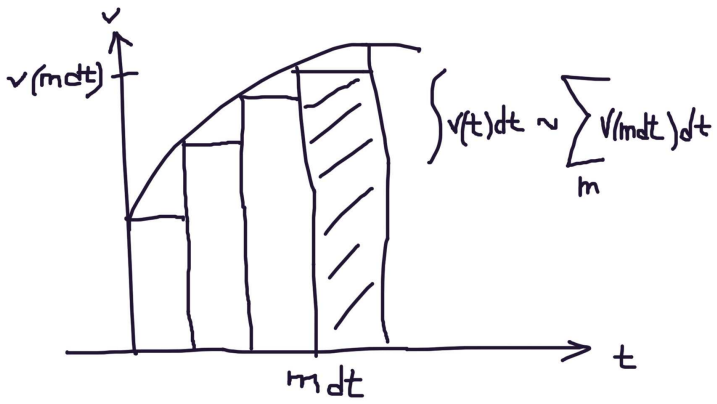

FIGURE 62.1. Integral as Riemann sum as area under graph.

62.2 Interpreting the Integral as an Area

The area $A(v,t)$ bounded by the graph of the function $v : [0,t] \to \mathbb{R}$ and the s-axis of a (v,s)-coordinate system, can viewed as a sum of rectangular strips of height $v(mdt)$ and width dt (asssuming for definiteness that $v(mdt) \geq 0$), and thus

$$A(v,t) = \sum_{m=0}^{n} v(mds)ds, \quad t = (n+1)ds. \tag{62.10}$$

We are thus led to interprete the integral as an area:

$$\int_0^t v(s)ds = A(v,t) = \text{ area under the graph of } v(s) \text{ on the interval } [0,t] \tag{62.11}$$

as illustrated in Fig. 60.1.

62.3 The Trapezoidal Rule

Replacing the shaded rectangle area in Fig. 60.1 with the area of a trapezoid right vertical of length $v((m+1)dt$ as illustrated in Fig. 60.2, we obtain the alternative Riemman sum approximation

$$\int_0^t v(s)ds \approx \sum_{m=0}^{n-1} \frac{v(mds) + v((m+1)ds)}{2}ds = \frac{v(0)}{2}ds + \sum_{m=1}^{n-1} v(mds)ds + \frac{v(t)}{2}ds. \tag{62.12}$$

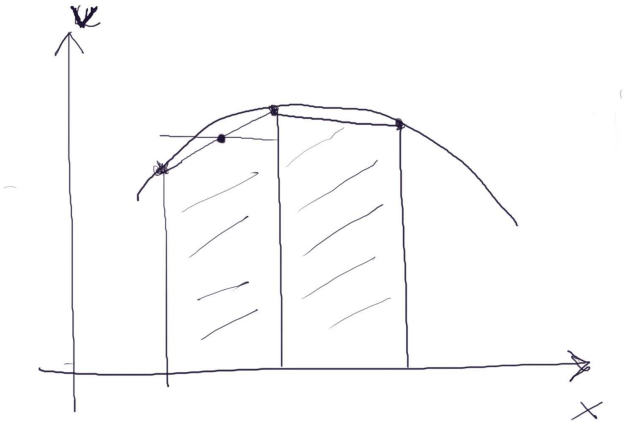

FIGURE 62.2. Piecewise linear approximation of the Trapezoidal Rule vs piecewise constant approximation of Euler Midpoint. Piecewise linear approximation is a basic element of computational mathematics including the finite elemeent method, as you will discover below...Simple and profound...

which is the *Trapezoidal Rule*. We compare with a *Midpoint Euler* method defining the height of the rectangle to be the function value at the midpoint of the interval. This value is close to the mean-value of the endpoint values used in the Trapezoidal Method, which thus is close to Midpoint Euler.

62.4 Not All Integrals are Areas

Note that distance is the integral of velocity but it is not very natural to say that distance is the area under the velocity graph.

Summing up: The integral is defined as the solution to an IVP. Some integrals can be interpreted as areas, but all integrals are not areas. Some cars (integrals) are Volvos (areas) but all cars (integrals) are not Volvos (areas). There are also Saabs...

Nevertheless, many Calculus books introduce the integral as the area under a graph, based on the pedagocial idea to define a new concept (the integral) in terms of something supposedly more familiar (area), but this is questionable from mathematical point of view and also confusing, when students discover that all integrals are not areas. To say that an integral is solution to an IVP, is not questionable, because this is what an integral *is*.

FIGURE 62.3. IVP of Usain Bolt

62.5 Watch

- Jesse Owens 1936 IVP: 100 on 10.3 sec
- Usain Bolt 2009 IVP: 100 m on 9.58 sec

63
The Fundamental Theorem of Calculus

63.1 Integration as Inverse of Differentation

The formula (62.7) is referred to as the *Fundamental Theorem of Calculus*: Integration of the function $v(t)$ followed by differentiation, gives back the function $v(t)$:

$$\frac{d}{dt}\int_0^t v(s)ds = v(t) \quad \text{for } t > 0. \tag{63.1}$$

Alternatively, The Fundamental Theorem of Calculus can be expressed as

$$\int_0^t \dot{u}(s)ds = u(t) \quad \text{for } t > 0, \tag{63.2}$$

stating: Integration of the derivative $\dot{u}(t)$ of the function $u(t)$, gives back the function $u(t)$. This follows from the fact that the derivative with respect to t of both sides of (63.2) equals $\dot{u}(t)$, combined with the fact that two functions with the same derivative taking the same value for $t = 0$, must coincide. Two cars traveling with the same velocity starting at the same time from the same location will arrive at the same time to the destination. Right?

We shall see that (63.2) can be viewed to express the following identity:

$$\text{The sum } (\int_0^t \text{ or } \sum_{m=0}^{n}) \quad \text{of the parts } (du = \dot{u}ds) \quad = \quad \text{the whole } (u(t)). \tag{63.3}$$

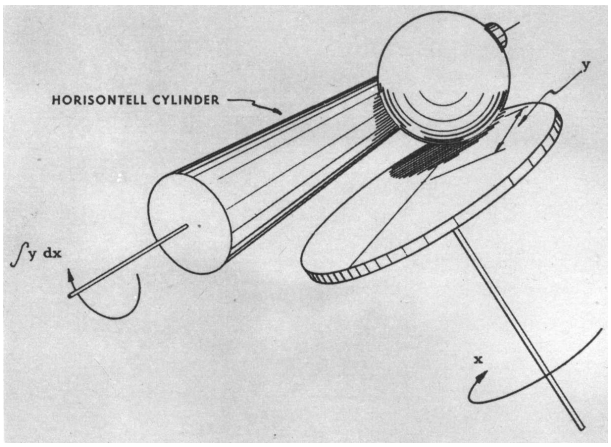

FIGURE 63.1. Analog mechanical integrator computing the integral $\int y(x)dx$ of a funtion $y(x)$. Can you explain how it works?

Integration means summing little pieces to make up the whole. In Leibniz notation this is expressed as

$$\int_0^t \frac{du}{ds}ds = \int_0^t du = u(t) - u(0). \tag{63.4}$$

Elementary and profound.

Below we shall study the dependence of the integral $\int_0^t v(s)ds$ of a given function $v(s)$ on the time step ds, and see that it is a uniquely determined number for vanishing time step, which is approximated using a finite time step, with accuracy depending on the variation of the function $f(t)$ with t. We will thus give a mathematical analysis of the meaning of the Fundamental Theoreom of Calculus, which we will refer to as a *mathematical proof* of the Fundamental Theorem.

This experience will illustrate the role and meaning of a mathematical proof as a process of dissecting the structure and meaning of a certain mathematical statement.

63.2 Read More

- Short Course in Calculus

- The Fundamental Theorem of Calculus

63.3 To Think About

- What could it mean to *prove* the Fundamental Theorem?
- Other interpretations of the whole = sum of parts?
- Suppose $u(T) = u(0) = 0$. What then about $\int_0^T u'(t)\,dt$?

63.4 Watch

- Babbages Difference Engine No. 2
- Leibniz binary ball computer
- $\sqrt{2}$ pepper grinder
- Kraftwerk Pocket Calculator
- Kraftwerk Numbers
- Computer World

> And as in arithmetic unpractised men must, and professors themselves may often, err, and cast up false; so also in any other subject of reasoning, the ablest, most attentive, and most practised men may deceive themselves, and infer false conclusions; not but that reason itself is always right reason, as well as arithmetic is a certain and infallible art: but no one man's reason, nor the reason of any one number of men, makes the certainty; no more than an account is therefore well cast up because a great many men have unanimously approved it. And therefore, as when there is a controversy in an account, the parties must by their own accord set up for right reason the reason of some arbitrator, or judge, to whose sentence they will both stand, or their controversy must either come to blows, or be undecided, for want of a right reason constituted by Nature; so is it also in all debates of what kind soever: and when men that think themselves wiser than all others clamour and demand right reason for judge, yet seek no more but that things should be determined by no other men's reason but their own, it is as intolerable in the society of men, as it is in play after trump is turned to use for trump on every occasion that suit whereof they have most in their hand. For they do nothing else, that will have every of their passions, as it comes to bear sway in them, to be taken for right reason, and that in their own controversies: bewraying their want of right reason by the claim they lay to it. (Leviathan, Thomas Hobbes)

75
Proof of the Fundamental Theorem

> The quadrature of all figures follow from the inverse method of tangents, and thus the whole science of sums and quadratures can be reduced to analysis, a thing nobody even had any hopes of before. (Leibniz)
>
> Knowing thus the Algorithm of this calculus, which I call Differential Calculus, all differential equations can be solved by a common method. (Leibniz)

Let us now study the effect of the time step in solution of

$$\dot{u}(t) = f(t), \quad \text{for } t > 0, \quad u(0) = u^0, \tag{75.1}$$

by Forward Euler time stepping

$$u(ndt + dt) = u(ndt) + f(ndt)dt, \quad n = 0, 1, 2, ... \tag{75.2}$$

We compare taking one step with time step dt with two steps of time step $\frac{dt}{2}$, for a given n:

$$\begin{aligned} u(ndt + dt) - \bar{u}(ndt + dt) &= f(ndt)dt - (f(ndt) + f(ndt + \frac{dt}{2}))\frac{dt}{2} \\ &= (f(ndt) - f(ndt + \frac{dt}{2}))\frac{dt}{2}, \end{aligned} \tag{75.3}$$

where $\bar{u}$ is computed with time step $\frac{dt}{2}$, and we assume that the same intial value for $t = ndt$ is used so that $\bar{u}(ndt) = u(ndt)$. Assuming that $f(t)$ is

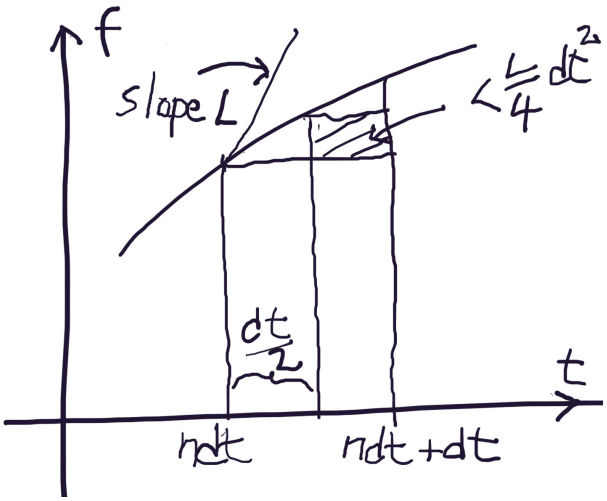

FIGURE 75.1. The fundamental step in the proof of the Fundamental Theorem.

Lipschitz continuous with Lipschitz constant L, we then find that

$$|u(ndt+dt)-\bar{u}(ndt+dt)| \leq \frac{L}{4}dt^2. \tag{75.4}$$

Summing now the contributions from all time steps with $n = 0, 1, 2, ..., N$, where $T = (N+1)dt$ is a final time, we get using that $\sum_{n=0}^{N} dt = T$,

$$|u(T)-\bar{u}(T)| \leq \frac{LT}{4}\,dt, \tag{75.5}$$

where thus $u(T)$ is computed with time step dt and $\bar{u}(T)$ with time step $\frac{dt}{2}$. Repeating the argument with successively refined times step $\frac{dt}{4}, \frac{dt}{8}, ...$, we get

$$|u(T)-\bar{u}(T)| \leq \frac{LT}{2}\,dt \tag{75.6}$$

for the difference between $u(T)$ computed with time step dt and $\bar{u}(T)$ computes with vanishingly small time step, since

$$\frac{1}{4}+\frac{1}{8}+\frac{1}{16}+... < \frac{1}{2}. \tag{75.7}$$

We have now proved the Fundamental Theorem of Calculus:

Theorem 75.1 *If $f : [0,T] \to \mathbb{R}$ is Lipschitz continuous, then the function $u(t) = \int_0^t f(s)$ defined by Forward Euler time-stepping with vanishing time step, solves the IVP: $\dot{u}(t) = f(t)$ for $t \in (0,1)$, $u(0) = 0$.*

The proof shows what it means *to understand* the Fundamental Theorem of Calculus, which means to realize that (letting k denote a finite time step and dt a vanishingly small step)

$$u(T) = \int_0^T f(t)\,dt \approx \sum_{n=0}^{N} f(nk)k \quad \text{if } T = (N+1)k, \tag{75.8}$$

as a consequence of

$$u((n+1)k) \approx u(nk) + f(nk)k, \quad \text{or} \quad \frac{u((nk+k) - u(nk)}{k} \approx f(nk), \tag{75.9}$$

where the sum is referred to as a *Riemann sum*, with the following bound for the difference

$$|\int_0^T f(t)\,dt - \sum_{n=0}^{N} f(nk)k| \leq \frac{LTk}{2} \quad \text{if } T = (N+1)k, \tag{75.10}$$

if $f : [0,T] \to \mathbb{R}$ is Lipschitz continuous with Lipschitz constant L.

In other words, *understanding* the integral $u(t) = \int_0^t f(s)\,ds$ of a function $f : [0,T] \to \mathbb{R}$ means to understand that it is determined by Riemann sums with vanishingly small step size, as the solution to the IVP $\dot{u}(t) = f(t)$, $u(0) = 0$, and to understand that the difference between two Riemann sums with mesh size k and $\frac{k}{2}$, is bounded by Lk (or more precisely by $\frac{L}{4}k$).

75.1 Even Better Understanding

As a serious student, you now probably ask: In precisely what sense the differential equation $\dot{u}(t) = f(t)$ is satisfied by an Euler Forward solution $u(t)$ with time step k? It certainly is so constructed, but can we get a direct verification? One way to do this is to associate a continuous piecewise linear function determined by the values $u(nk)$ at the discrete time levels nk, again denoted by $u(t)$. We then have on each interval $(nk,(n+1)k)$, by the definition of $u(t)$:

$$\dot{u}(t) = \frac{u((n+1)k) - u(nk)}{k} = f(nk), \tag{75.11}$$

from which we conclude that

$$|\dot{u}(t) - f(t)| \leq |f(nk) - f(t)| \leq Lk \quad \text{for } t \in ((n+1)k, nk). \tag{75.12}$$

We can thus say that $u(t)$ satisfies the differential equation $\dot{u}(t) = f(t)$ for all t with a precision of Lk. In other words, the *residual* $\dot{u}(t) - f(t)$ is smaller than Lk. We have now understood the Fundamental Theorem even better, right?

FIGURE 75.2. The sad result of Archimedes mathematics.

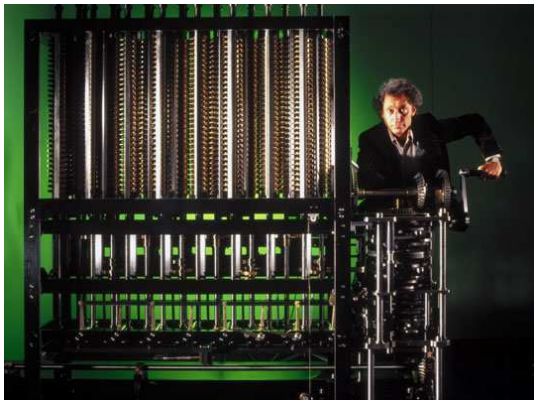

FIGURE 75.3. Babbage's Difference Engine No. 2 1847.

We shall see below that extending a function defined on a discrete set of points to a continuous piecewise linear function, is a central aspect of approximation in general and of the finite element method in particular.

75.2 To Think About

- What is fundamental about the Fundamental Theorem?
- Why is $\frac{d}{dt}\int_0^t f(s)\,ds = f(t)$? (compare with last argument)
- What is the Riemann sum error using the Trapezoidal Rule (62.12)?

Hint: $\int_0^{t+dt} f(s)\,ds - \int_0^t f(s)\,ds = \int_t^{t+dt} f(s)\,ds = f(t)dt \pm \frac{L}{2}dt^2$.

76
Contraction Mapping for $u = g(u)$

Give me a fixed point, and I will move the Earth. (Archimedes)

76.1 Solving $f(u) = 0$ by Time Stepping

To solve an equation $f(u) = (f_1(u), f_2(u), ..., f_N(u)) = 0$ of N equations $f_i(u) = 0$, $i = 1, ..., N$, in N unknowns $u = (u_1, u_2, ..., u_N)$, with thus $f : \mathbb{R}^n \to \mathbb{R}^n$, it is natural to connect to solution of the IVP: Find $u(t)$ such that

$$\dot{u}(t) + f(u(t)) = 0 \quad \text{for } t > 0, \quad u(0) = u^0, \tag{76.1}$$

with some given initial value u^0. If it turns out that as t increases, the function $u(t)$ tends to some value $\hat{u}$, then $\dot{u}(t)$ could be expected to become small, and if so, we would have

$$f(u(t)) \approx 0. \tag{76.2}$$

and we would be led to set $\hat{u} = u(t)$ for some large t and consider $\hat{u}$ to be an approximate solution of $f(u) = 0$ with small residual $f(\hat{u})$.

If $f(u)$ has several different solutions, which is often the case, then we could expect to capture different solutions by chosing different initial values u^0.

Computing $u(t)$ by Forward Euler with time step $dt = 1$, we would have

$$u^{n+1} = u^n - f(u^n), \quad for\ n = 0, 1, 2, ..., \tag{76.3}$$

If $|u^{n+1} - u^n|$ would become small for increasing n, then $f(u^n)$ would become small and thus u^n would be an approximate solution of $f(u) = 0$ with small residual $f(u^n)$.

76.2 Solving $u = g(u)$

We are thus led to study the convergence of the iteration

$$u^{n+1} = g(u^n), \quad n = 0, 1, 2, ..., \tag{76.4}$$

where

$$g(u) = u - f(u). \tag{76.5}$$

To this end we take the difference of (76.4) for two consecutive steps to get

$$e^{n+1} \equiv u^{n+1} - u^n = g(u^n) - g(u^{n-1}). \tag{76.6}$$

If $g : \mathbb{R}^N \to \mathbb{R}^N$ is Lipschitz continuous with Lipschitz constant L, then

$$|e^{n+1}| = |g(u^n) - g(u^{n-1})| \leq L|e^n| \leq L^2|e^{n-1}| \leq L^n|u^1 - u^0| \tag{76.7}$$

We see that if $L < 1$, then $|e^n|$ becomes vanishingly small as n increases, which by (76.3) means that $f(u^n)$ becomes vanishingly small and thus u^n may be viewed as an approximate solution of $f(u)$ in the sense that the residual $f(u^n)$ is small. In the next chapter we also consider the error in the approximate root u^n.

We see that if $L << 1$ then the convergence is fast, and if $L \approx 1$ then the convergence is slow. If $L = \frac{1}{2}$ then the residual $|g(u^n) - u^n| = |u^{n+1} - u^n|$ is reduced with a factor 2 in each iteration step, that is with a binary digit per step.

If $L < 1$ then the mapping $u \to g(u)$ is said to be a *contraction*, because

$$|g(u) - g(v)| \leq L|u - v| < |u - v| \tag{76.8}$$

expressing that the distance between the images $|g(u) - g(v)|$ is smaller than the distance between the arguments $|u - v|$. We have just proved the famous

Contraction Mapping Theorem: If $g : \mathbb{R}^N \to \mathbb{R}^N$ is a contraction with Lipschitz constant $L < 1$, then the iteration $u^{n+1} = g(u^n)$ converges to a unique fixed point satisfying $u = g(u)$ at the rate L^n.

77
Newton's Method for $f(u) = 0$

> The sciences, are small power; because not eminent; and therefore, not acknowledged in any man; nor are at all, but in a few; and in them, but of few things. For science is of that nature, as none can understand it to be, but such as in a good measure have attained it. (Thomas Hobbes in Leviathan Chapter X 14.)
>
> Arts of public use, as fortifications, making of engines, and other instruments of war; because they confer to defence, and victory, are power: and though the true mother of them, be science, namely the mathematics; yet, because they are brought into the light, by hand of the artificer, they be esteemed (the midwife passing with vulgar for the mother,) as his issue. (Thomas Hobbes in Leviathan Chapter X 15.)

We now consider a variant of (76.3) for solving $f(u) = 0$ with faster convergence by invoking the (inverse of the) derivative $f'(u)$, referred to as *Newton's Method*.

Let us then start with $N = 1$ and let $f : \mathbb{R} \to \mathbb{R}$ be a differentiable function. Consider the following iteration:

$$u^{n+1} = u^n - \frac{f(u^n)}{f'(u^n)} = g(u^n) \tag{77.1}$$

with corresponding function

$$g(u) = u - \frac{f(u)}{f'(u)} \tag{77.2}$$

assuming that $f'(u) \neq 0$. Computing the derivative $g'(u)$, we get

$$g'(u) = 1 - \frac{f'(u)}{f'(u)} + \frac{f(u)f''(u)}{(f'(u))^2} = 0, \tag{77.3}$$

if $f(u) = 0$. Thus we may expect that $|g'(u)|$ is small, that is that $L << 1$ implying fast convcergence.

The iteration (254.1) is called *Newton's Method* for computing a solution of the equation $f(u) = 0$. Newton's method directly generalizes to $f : \mathbb{R}^N \to \mathbb{R}^N$ in the form

$$u^{n+1} = u^n - (f'(u^n))^{-1} f(u^n) \tag{77.4}$$

where $f'(u^n))^{-1}$ is the inverse of the $N \times N$ matrix $f'(u^n)$ (thus assuming that $f'(u^n)$ is non-singular). One can show that $|e^{n+1}| \sim |e^n|^2$, if the initial guess is close enough to the root, which means that the number of correct digits may double at iteration step.

77.1 Wellposed and Illposed Roots

Suppose u is an approximate solution with residual $f(u) \approx 0$, or approximate *root*, of an equation $f(u) = 0$ with exact root $\bar{u}$. We have for small $|u - \bar{u}|$

$$f(u) - f(\bar{u}) \approx f'(u)(u - \bar{u}), \tag{77.5}$$

(still assuming for simplicity $N = 1$. This shows that

$$|u - \bar{u}| \approx \frac{|f(u)|}{|f'(u)|} \tag{77.6}$$

indicating that the residual error $|f(u)|$ translates to the root error $|u - \bar{u}|$ with the *stability factor*

$$S = \frac{1}{|f'(u)|}, \tag{77.7}$$

that is

$$|u - \bar{u}| \approx S|f(u)| \tag{77.8}$$

In other words: If $|f'(u)|$ is not small so that S is not large, then the root is well defined or *wellposed*, while if $|f'(u)|$ is small so that S is large, then the root is *illposed* or not well defined.

For a wellposed root u the curve $x \to f(x)$ crosses the x-axis at $x = u$ with a definite slope, which makes the crossing point well determined. For an illposed root the curve is almost tangent to the x-axis which makes the crossing point difficult to pin down.

77.2 Newton's Method Requires Good Initial Guess

Newton's method converges very quickly towards a root, if the starting value is close enough to the root. If not, the iterations may diverge and then give rise complex fractal patterns as shown in the figure below showing big basins of convergence around roots separated by fractal boundary zones.

77.3 Learn More

- Fixed point iteration.
- Newton's method

77.4 To Think About

- How to compute $\sqrt{2}$? By Solving $x^2 = 2$? How?

77.5 Watch

- Newton's method fractal 1
- Newton's method fractal 2
- Newton fractals algorithm

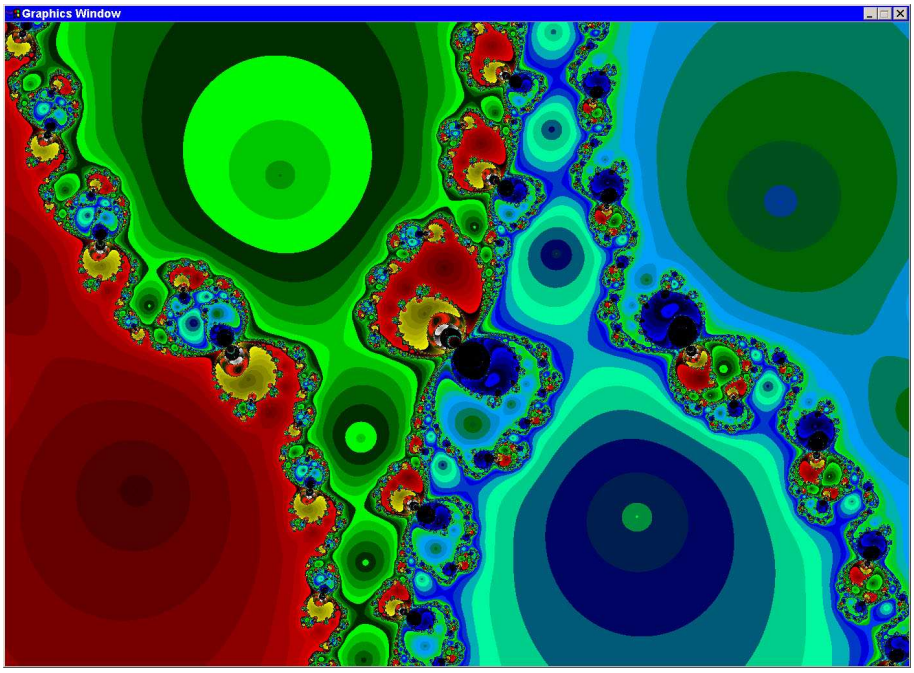

FIGURE 77.1. Fractals from iterations by Newton's method. Big basins show roots. Boundaries between basins show fractal complexity.

78
Generalized Fundamental Theorem

I believe in the fundamental Truth of all the great religions of the world. I believe that they are all God given. I came to the conclusion long ago... that all religions were true and also that all had some error in them. (Mahatma Gandhi)

The fairest thing we can experience is the mysterious. It is the fundamental emotion which stands at the cradle of true art and true science. He who know it not and can no longer wonder, no longer feel amazement, is as good as dead, a snuffed-out can (Einstein)

Most of the fundamental ideas of science are essentially simple, and may, as a rule, be expressed in a language comprehensible to everyone. (Einstein)

78.1 Time Stepping $\dot{u} = u$

The Fundamental Theorem concerns time-stepping of the IVP

$$\dot{u}(t) = f(t) \quad \text{for } t > 0, \quad u(0) = u^0, \tag{78.1}$$

where the Lipschitz continuous function $f(t)$ does not depend on the unknown u, only on the (independent) time variable t.

We now extend to allow f to depend also on u. Assuming for simplicity no explicit dependence on t, we thus consider the IVP:

$$\dot{u}(t) = f(u(t)) \quad \text{for } t > 0, \quad u(0) = u^0, \tag{78.2}$$

where $f : \mathbb{R} \to \mathbb{R}$ is a given Lipschitz continuous function with Lipschitz constant L. The basic question is if the solution can be computed to arbitrary precision by time-stepping?

The basic case is $f(u) = u$ and $u^0 = 1$, that is the IVP:

$$\dot{u}(t) = u(t) \quad \text{for } t > 0, \quad u(0) = 1, \tag{78.3}$$

with $L = 1$ and the solution $u(t) = \exp(t)$ computed by Forward Euler:

$$\exp(t) \approx (1 + \frac{t}{n})^n \tag{78.4}$$

with time step $k = \frac{t}{n}$. We estimate the effect of dividing the time-step by a factor 2, using that $(t + dt)^n - t^n \approx nt^{n-1}dt$ (because $\frac{d}{dt}t^n = nt^{n-1}$):

$$\begin{aligned}(1 + \frac{t}{2n})^{2n} - (1 + \frac{t}{n})^n &= ((1 + \frac{t}{2n})(1 + \frac{t}{2n}))^n - (1 + \frac{t}{n})^n \\ &= ((1 + \frac{t}{n} + \frac{t^2}{4n^2})^n - (1 + \frac{t}{n})^n \approx n(1 + \frac{t}{n})^{n-1}\frac{t^2}{4n^2} \approx \frac{t}{n}\exp(t)\frac{t}{4}.\end{aligned}$$

We see that the difference is proportional to the time step $k = \frac{t}{n}$. As in the proof of the Fundamental Theorem of Calculus, we conclude that $(1 + \frac{t}{n})^n$ determines $\exp(t)$ with a precision proportional to the time step with a multiplicative factor $\approx \exp(t)\frac{t}{4} \sim \exp(t)$.

78.2 Time Stepping $\dot{u} = f(u)$

The above proof extends to an arbitrary Lipschitz continuous function $f(u)$ with the difference that the time-stepping error in computing $u(t)$, now is proportional to the time step with a multiplicative factor $\exp(Lt)$, where L is the Lipschitz constant of f. This extends to systems with $f : \mathbb{R}^d \to \mathbb{R}^d$ with $d > 1$.

It is natural to refer to this result as a *Generalized Fundamental Theorem*: The Fundamental Theorem concerns $\dot{u}(t) = f(t)$ and the Generalized Fundamental Theorem concerns $\dot{u}(t) = f(u(t))$. Calculus in a nutshell!

A proof of the Generalized Fundamental Theorem can be performed by combining the following two steps:

Step 1: Estimate the difference $u((n + 1)k) - \tilde{u}((n + 1)k)$ by taking one (Forward Euler) time step of length k and two time steps on length $\frac{k}{2}$, from

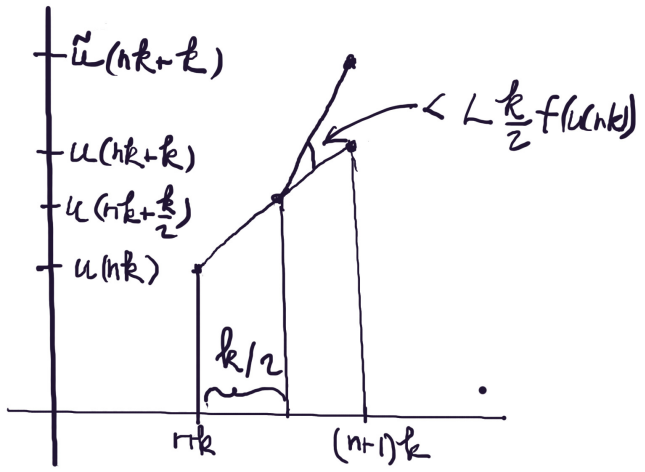

FIGURE 78.1. Fundamental Step 1 in the proof of the Generalized Fundamental Theorem.

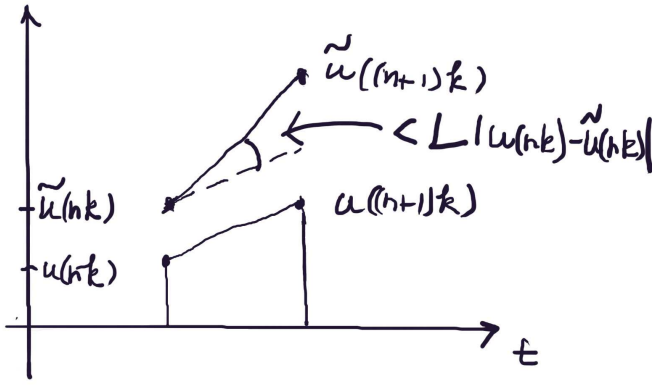

FIGURE 78.2. Fundamental Step 2 in the proof of the Generalized Fundamental Theorem.

the same initial value $u(nk)$:

$$\begin{aligned}
&|u((n+1)k) - \tilde{u}((n+1)k)| \\
&= |u(nk) + kf(u(nk)) - (u(nk) + \frac{k}{2}f(u(nk)) + \frac{k}{2}f(u(nk) + \frac{k}{2}f(u(nk)))| \\
&= \frac{k}{2}|f(u(nk) - f(u(nk) + \frac{k}{2}f(u(nk)))| \leq \frac{k}{2}L\frac{k}{2}|f(u(nk))|.
\end{aligned} \tag{78.5}$$

Step 2: Estimate the difference after one time step from different initial conditions $u(nk) - \tilde{u}(nk)$:

$$\begin{aligned}
|u((n+1)k) - \tilde{u}((n+1)k) &= |u(nk) + kf(u(nk)) - \tilde{u}(nk) + kf(\tilde{u}(nk))| \\
&\leq (1 + kL)|u(nk) - \tilde{u}(nk)|.
\end{aligned} \tag{78.6}$$

Combining Steps 1 and 2, we obtain a final error proportional to the time step k with a multiplicative factor $\exp(Lt)$, which we refer to as a *stability factor*. For details see Completion of the Proof.

To see the connection with the basic case $f(u) = u$, think of estimating a general function $f(u)$, assuming $f(0) = 0$ for simlipicity, by $f(u) \approx f'(0)u$, which suggests that the factor $\exp(t)$ for $f(u) = u$ should be replaced by $\exp(Lt)$ for a general $f(u)$ (because $|f'(0)| \leq L$).

Generalization to a vector valued function $f : \mathbb{R}^d \to \mathbb{R}^d$ with $d > 1$ is direct, and we have thus presented the essential steps of a proof of the following main result of Calculus:

Theorem 78.1 Generalized Fundamental Theorem of Calculus: *The solution $u(t)$ of the IVP $\dot{u}(t) = f(u(t))$ for $0 < t \leq T$ with $u(0)$ given, where $f : \mathbb{R}^d \to \mathbb{R}^d$ is Lipschitz continuous with Lipschitz constant L, is uniquely computable by Forward Euler time stepping with a precision proportional to the time step times a stability factor of size* $\exp(LT)$.

For a completion of the proof, see below and Special Case and General Case.

The proof is similar for the other methods we have so far encountered (with $u^n = u(nk)$):

$$\begin{aligned}
u^{n+1} &= u^n + kf(u^{n+1}) \quad \text{Backward Euler,} \\
u^{n+1} &= u^n + kf(\frac{u^n + u^{n+1}}{2}) \quad \text{Midpoint Euler,} \\
u^{n+1} &= u^n + \frac{k}{2}(f(u^n) + f(u^{n+1})) \quad \text{Trapezoidal Method.}
\end{aligned} \tag{78.7}$$

We see that if $f(u)$ is linear in u, then Midpoint Euler and the Trapezoidal Method coincide.

78.3 A Posteriori Error Control

For a more precise error control, based on computed solutions, see

- Time Stepping Error Analysis
- Time Stepping by FEM

78.4 The Illusion of an $\exp(LT)$ Bound

If $L = 10$ and $T = 30$, which looks pretty harmeless, then $\exp(LT) = \exp(300) >> 10^{100} =$ **googol**, an incredibly large number, much larger than the number of atoms in the Universe. A matching time step of 10^{-100} is beyond all rationale and thus computation of a solution of an IVP with moderate Lipschitz constant oevr a moderatley long time interval may be impossible. An example is the Lorenz system with

$$f(u) = (-10u_1 + 10u_2, 28u_1 - u_2 - u_1u_3, -\frac{8}{3}u_3 + u_1u_2), \tag{78.8}$$

for which computation on an interval of length T requires computation with about $T/2$ digits, see:

- BS The Lorenz System and the Essence of Chaos
- Long-Time Computability of the Lorenz System

78.5 Stiff IVPs

There is a class of IVPs with large or very large Lipschitz constants, which are computable on long time intervals, because the function $f(u)$ has a decay property (negative derivative) causing errors to decay rather than grow exponentially. Such problems are called *stiff problems* and may require implicit time stepping to avoid severe time step restrcitions in explicit methods. See Stiff Problems.

78.6 Wave Equations

IVPs with wavelike solutions, like the system

$$\dot{u}_1 = u_2 \quad \dot{u}_2 = -u_1 \tag{78.9}$$

with solutions being linear combinations of $\sin(t)$ and $\cos(t)$, have formally Lipschitz constants of size 1, can be integrated with error growth $\sim t$, instead of $\sim \exp(t)$ by the above (crude) estimate, if a proper time-stepping

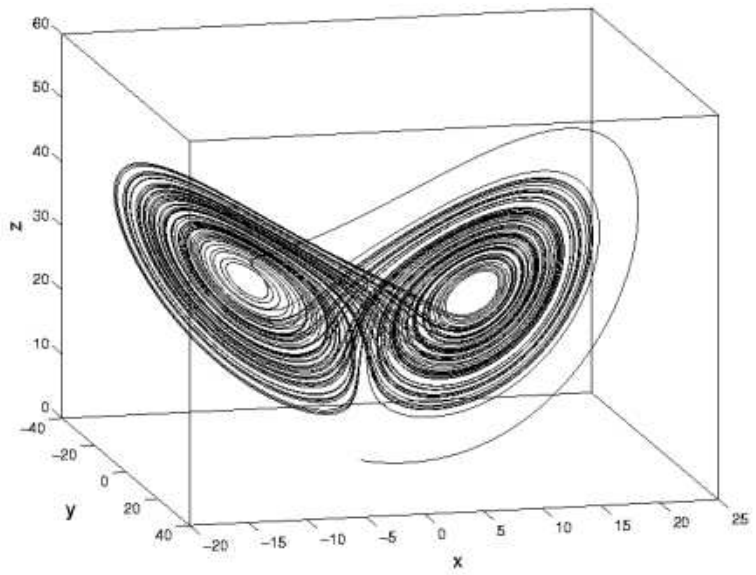

FIGURE 78.3. A Lorenz system solution trajectory.

method (like cG(1)) is used. This is due to error cancellation in wave motion.

For more complex wave problems, or problems with more or less periodic solutions, the stability factor can have a polynomial growth in time t, e.g. quadratic for simple planetary systems.

78.7 Summary: Time Stepping of IVP

The precision in time stepping the solution $u(t)$ of an IVP $\dot{u} = f(u)$ for $0 < t \leq T$, with first order method with time step k, can be estimated by $S(T)k$, where $S(t)$ acts as stability factor measuring error propagation and accumulation of size

- $S(T) \sim \exp(LT)$ (general), where L is the Lipschitz constant of f.
- $S(T) \sim 1$ (stiff: diffusion problems)
- $S(T) \sim T$ (wave problems), $\quad S(T) \sim T^2$ (planetary system).

78.8 Preparing for a More Precise Analysis

As a preparation for the more precise error analysis in Time Stepping Error Analysis and Time Stepping by FEM, we consider two solutions $u(t)$ and $\tilde{u}(t)$ com-

puted with the same time step k but different initial data u^0 and $\tilde{u}^0$. Subtracting the update formulas we have formally for the difference $e = u - \tilde{u}$:

$$\dot{e}(t) \approx f'(u(t))e(t) \quad \text{for } t > 0, \quad e(0) = e^0 \equiv u^0 - \tilde{u}^0 \tag{78.10}$$

showng that an initial error is propagated as a solution to a linearized IVP with coefficient $f'(u(t))$ depending on a computed solution $u(t)$. We shall see that by solving the linearized problem (or rather a closely related dual linearized problem), the stability factors $S(t)$ measuring error growth can be computed and the precision on the computation of $u(t)$ can be assessed.

The linearized problem (or its dual) thus gives the key to unlock time stepping precision.

Note that with $f(u) = u$, the linearized problem reads $\dot{e} = e$ with solution $e(t) = \exp(t)e^0$, showing exponential error growth, as expected.

78.9 Completion of the Proof

To complete the proof of the Generalized Fundmental Theorem we are to sum up the error contributions from each subinterval of length k, which according to Step 1 and Step 2 amounts to

$$\begin{aligned}\sum_{n=1}^{N}(1+kL)^n Lk^2M &\approx kML\sum_{n=0}^{N}\exp(Lnk)k \\ &\approx kML\int_0^T \exp(Ls)\,ds \approx kM\exp(LT),\end{aligned} \tag{78.11}$$

where $Nk = T$ and $M \geq \max_u |f(u)|$. Can you explain what is going on here? If not, take a look at:

78.10 Hint to Completion of the Proof

We can think of comparing computations with k and $\frac{k}{2}$ with corresponding solutions $u(t)$ and $\tilde{u}(t)$ in two ways depending on how we choose initial values on each time interval $(nk, (n+1)k)$:

1. Compute $u(t)$ and $\tilde{u}(t)$ independently with timestep k and $\frac{k}{2}$.
2. Assume that $\tilde{u}(nk) = u(nk)$ and account for the effect at final time of the difference $\tilde{u}(nk) - u(nk)$.

1. is the most direct from computational point of view and a corresponding proof is given in General Case.

Here we consider 2. because the proof is (maybe) simpler: The error from the first time step is bounded by Lk^2M assuming no error in initial data,

and is propagated with a factor bounded by $(1 + kL)$ for each time step, thus with a factor $(1 + kL)^N$ after N steps. Similarly, the error from the second time step is bounded by Lk^2M, again assuming no error in the corresponding initial value, and is propagated with a factor $(1 + kL)^{N-1}$, et cet. Summing we obtain a bound of the total error after N steps.

It is instructive to in illustrate 1. and 2. in a figure complementing Figs. 77.1-2.

78.11 Uniqueness of Solution

To prove that the solution of the IVP (**??**) is unique, we assume $v(t)$ is a possibly different solution also satisfying $\dot{v}(t) = f(v(t))$ for $t > 0$ and $v(0) = u^0$. Subtraction gives for the difference $w = u - v$

$$\dot{w} = f(u) - f(v) \tag{78.12}$$

and thus taking the scalar product with w and using Cauchy's inequality, we get

$$\frac{d}{dt}\frac{1}{2}|w|^2 = \frac{d}{dt}\frac{1}{2}w \cdot w = (f(u) - f(v)) \cdot w \leq L|w||w| \tag{78.13}$$

and thus for $W = |w|^2$

$$\dot{W} \leq 2LW, \tag{78.14}$$

which shows that (why?)

$$W(t) \leq W(0)\exp(2Lt) \quad \text{for } t > 0. \tag{78.15}$$

But $W(0) = |u(0) - v(0)| = 0$ and thus $W(t) = 0$ and $u(t) = v(t)$ for $t > 0$ and uniqueness follows. But to be scientifically honest, size of the exponential factor $\exp(Lt)$ is crucial. If $L10$ and $t = 30$, which does not look too frightening, then $\exp(Lt) = \exp(300) > 10^{100}$ =googol, which means that the argument the $\exp(300)W(0)$ is small (zer0) if $W(0)$ is small (zero), is questionable, very questionable, right?

78.12 How to Prove $\exp(t + s) = \exp(t)\exp(s)$?

To prove the basic law of the exponential $\exp(t + s) = \exp(t)\exp(s)$, note that the function $u(s) = \exp(t + s)$ satisfies $\frac{du}{ds} = u$ for $s > 0$ and $u(0) = \exp(t)$. But the function $v(s) = \exp(t)\exp(s)$ also satisfies $\frac{dv}{ds} = v$ for $s > 0$ and $v(0) = \exp(t)$, and by uniqueness $u(s) = v(s)$, that is $\exp(t + s) = \exp(t)\exp(s)$.

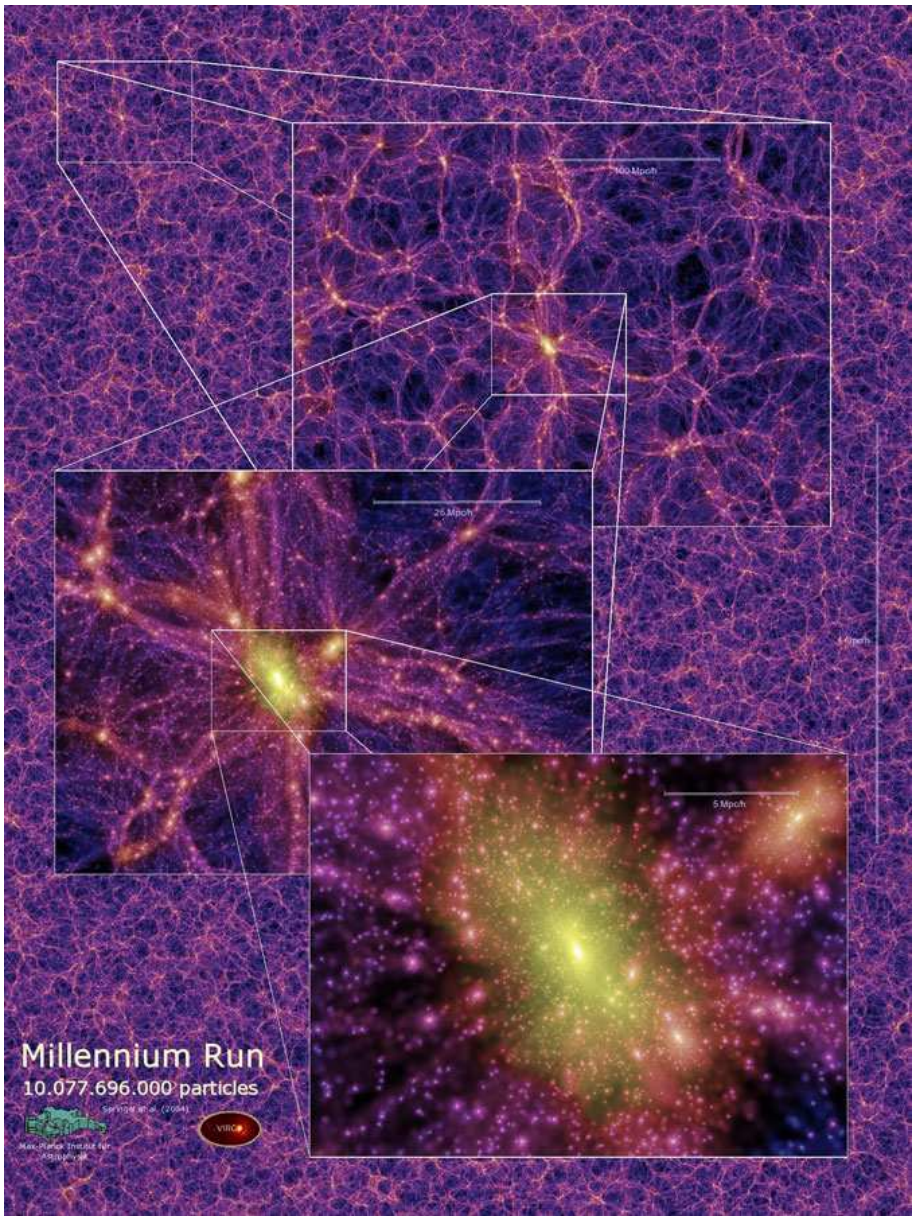

FIGURE 78.4. The Millennium Run: A large n-body computation with $n = 10.077.696.000$.

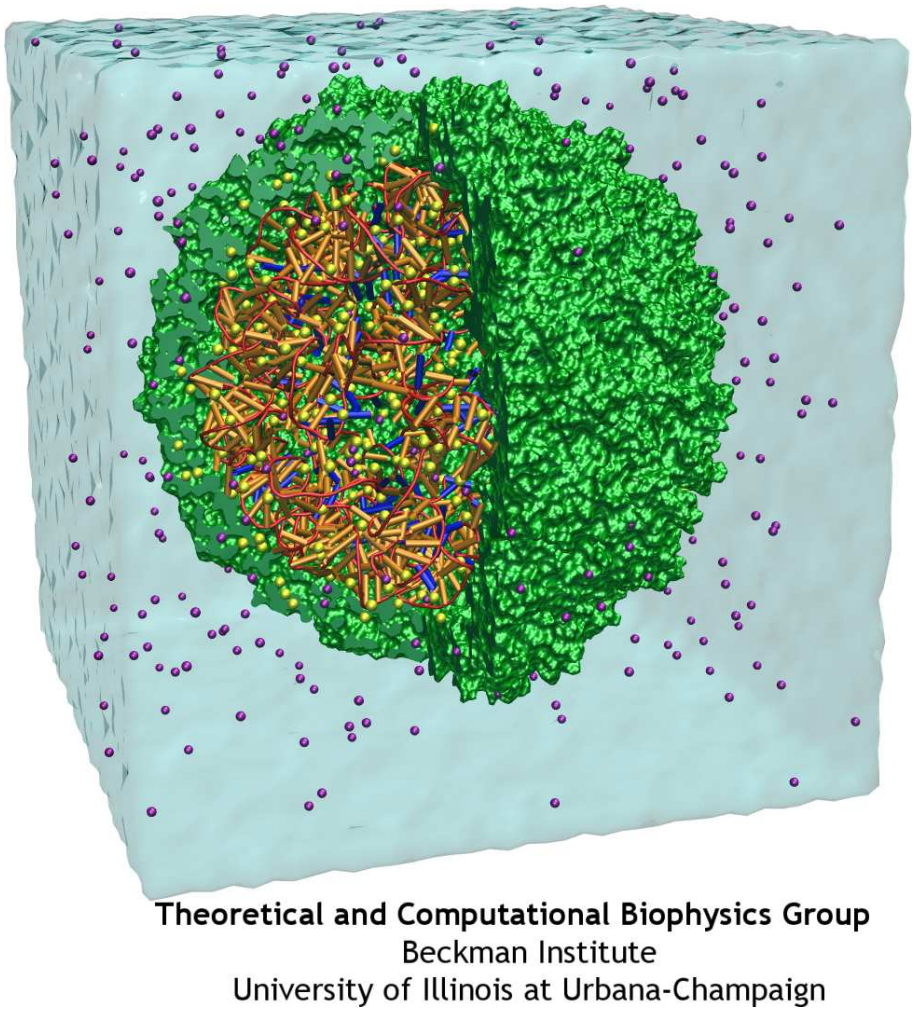

FIGURE 78.5. Model of a virus as a large molecular dynamics system of differential equations $\dot{u} = f(u)$.

81
Time Stepping Error Analysis

Sapiens nihil affirmat quod non probat.

After experience had taught me that all the usual surroundings of social life are vain and futile; seeing that none of the objects of my fears contained in themselves anything either good or bad, except in so far as the mind is affected by them, I finally resolved to inquire whether there might be some real good having power to communicate itself, which would affect the mind singly, to the exclusion of all else: whether, in fact, there might be anything of which the discovery and attainment would enable me to enjoy continuous, supreme, and unending happiness. (Spinoza)

81.1 Midpoint Euler

Consider a linear scalar IVP of the form: Find $u(t)$ such that

$$\dot{u}(t) + Au(t) = F(t) \quad \text{for } t > 0, \quad u(0) = u^0, \tag{81.1}$$

where A is a constant and $F(t)$ a given functions, which has the standard form $\dot{u}(t) = f(t, u(t))$ with $f(t, u) = F(t) - Au$.

Compute an approximate solution $U(t)$ by time stepping according to the Trapezoidal Method: Find $U^n = U(nk)$ such that

$$U^{n+1} + \frac{k}{2}(AU^{n+1} + AU^n) = U^n + \int_{I_n} F(t)\,dt, \quad \text{for } n = 0, 1, 2, ..., \tag{81.2}$$

with $I_n = (nk, (n+1)k)$ and $U^0 \approx u^0$, assuming the integral of $F(t)$ can be evaluated analytically. We here think of $U(t)$ as a piecewise linear continuous function taking on the computed values U^n at the discrete time levels nk with time step k. If so the Trapezoidal Method and Midpoint Euler coincide. In particular, we then have

$$\int_{I_n} \dot{U}\, dt = U^{n+1} - U^n, \quad \int_{I_n} AU\, dt = \frac{k}{2}(AU^{n+1} + AU^n), \tag{81.3}$$

which shows that Trapezoidal/Midpoint Euler satisfies:

$$\int_{I_n} (\dot{U} + AU - F)\, dt = 0 \quad \text{for } n = 0, 1, 2, ... \tag{81.4}$$

In other words, the mean-value over each subinterval I_n of the *residual* $R(U) \equiv \dot{U} + AU - F$ vanishes.

We shall consider the following basic choices of A with different stability characteristics connecting back to Summary: Timestepping of IVP:

1. constant non-negative: $A \geq 0$,
2. constant imaginary: $A = i$,
3. constant negative: $A < 0$,
4. oscillating positive-negative: $A(t) = \sin(t)$,

where we also added a basic case with $A(t)$ depending on t. We shall below see that (81.1) can also be interpreted as a system of differential equations with A a square matrix with the following analogous stability characteristics:

1. A positive semi-definite (symmetric): diffusion problems
2. A anti-symmetric: wave problems
3. A negative definite (symmetric) or general matrix: inverted pendulum...
4. A oscillating with zero mean: turbulence...

For a general non-linear system $\dot{u}+f(u) = 0$, the matrix A then corresponds to the Jacobian $f'(u(t))$ as concerns stability.

81.2 Error Analysis of Midpoint Euler

We shall now estimate the error $|u(T) - U(T)|$ at a given time $T = Nk$ in terms of the time step k and relevant quantities to be defined, where

we assume the $u(t)$ is an exact solution satisfying (81.1) to high precision (computed with a (vanishingly) small time step).

We shall do this by representing the error in terms of the solution $\varphi(t)$ of the following *dual problem*:

$$\begin{cases} -\dot{\varphi} + A\varphi = 0 & \text{for } T > t \geq 0, \\ \varphi(T) = \pm 1 = \text{sign of } e(T), \end{cases} \tag{81.5}$$

where $e = u - U$. We note that that (81.5) runs "backwards in time" starting at time T, because the (initial) data is given at $t = T$, and that (accordingly) the time derivative term $\dot{\varphi}$ has a minus sign. We start from the identity

$$0 = \int_0^T e\,(-\dot{\varphi} + A\varphi)\,dt,$$

and integrate by parts to get the following error representation (since $|e(T)| = e(T)\varphi(T)$):

$$|e(T)| = \int_0^T (\dot{e} + Ae)\varphi\,dt + e(0)\varphi(0),$$

where we allow $U(0)$ to be different from $u(0)$, corresponding to an error $e(0)$ in the initial value $u(0)$. Since u solves the differential equation (159.14), that is $\dot{u} + Au = F$, we have

$$\dot{e} + Ae = \dot{u} + Au - F - (\dot{U} + AU - F) = F - \dot{U} - AU = -R(U),$$

and thus we obtain the following representation of the error $|e(T)|$ in terms of the residual $R(U) = \dot{U} + AU - F$ and the dual solution φ:

$$|e(T)| = -\int_0^T R(U)\varphi\,dt + e(0)\varphi(0). \tag{81.6}$$

Recalling (81.4) we have

$$\int_{I_n} R(U)\,dt = 0 \quad \text{for } n = 0, 1, 2, ...,$$

which allows us to rewrite (81.6) as

$$|e(T)| = -\int_0^T R(U)(\varphi - \bar{\varphi})\,dt + e(0)\varphi(0), \tag{81.7}$$

where $\bar{\varphi}$ is the mean-value of φ over each time interval I_n, that is

$$\bar{\varphi}(t) = \frac{1}{k}\int_{I_n} \varphi(s)\,ds \quad \text{for } t \in I_n.$$

We shall now use the fact that

$$\int_{I_n} |\varphi - \bar{\varphi}|\, dt \le k \int_{I_n} |\dot{\varphi}|\, dt,$$

which follows by integration from the facts that

$$\varphi(t) - \bar{\varphi}(t) = \frac{1}{k} \int_{I_n} (\varphi(t) - \varphi(s))\, ds,$$

and

$$|\varphi(t) - \varphi(s)| \le \int_s^t |\dot{\varphi}(\sigma)|\, d\sigma \le \int_{I_n} |\dot{\varphi}(\sigma)|\, d\sigma \quad \text{for } s, t \in I_n.$$

Thus, (159.17) implies

$$\begin{aligned} |e(T)| &\le \sum_{n=0}^{N-1} R_n \int_{I_n} |\varphi - \bar{\varphi}| dt + |e(0)||\varphi(0)| \\ &\le \sum_{n=0}^{N-1} k\, R_n \int_{I_n} |\dot{\varphi}| dt + |e(0)||\varphi(0)|, \end{aligned} \tag{81.8}$$

where

$$R_n(U) = \max_{t \in I_n} |R(U(t))|.$$

Bringing out the max of $k_n R_n$ over n, we get

$$|e(T)| \le \max_{0 \le n \le N-1} k R_n \int_0^T |\dot{\varphi}|\, dt + |e(0)||\varphi(0)|.$$

Defining the *stability factors* $S_c(T)$ and $S_d(T)$ by

$$S_c(T) = \int_0^T |\dot{\varphi}(s)|\, ds, \quad S_d(T) = |\varphi(0)|, \tag{81.9}$$

we get the following *a posteriori error estimate*:

Theorem 81.1 *The approximate solution $U(t)$ of the initial value problem (81.1) computed by Midpoint Euler with time step k over intervals I_n, satisfies for $T > 0$*

$$|u(T) - U(T)| \le S_c(T) \max_n k R_n(U) + S_d(T)|u(0) - U(0)|, \tag{81.10}$$

where $u(t)$ is the exact solution computed with vanishingly small time step, $R_n(U) = \max_{I_n} |\dot{U} + AU - F|$ measures the residual over I_n, and $S_c(T)$ and $S_c(T)$ are stability factors defined by (81.9) related to time-discretization and initial data.

The stability factors $S_c(T)$ and $S_d(T)$ measure the effects of the accumulation of error in the approximation. To give the analysis a quantitative meaning, we have to give a quantitative bound of these factors. In general the stablity factors are computed by computing the solution of the dual problem. In special cases the stability factors can be computed analytically, as we now show:

The following lemma gives an estimate for $S_c(T)$ and $S_d(T)$ depending on the nature of A, in particular the sign of A, with possibly vastly different stability factors. We notice that the solution $\varphi(t)$ of (159.15) if A is constant is given by the explicit formula

$$\varphi(t) = \pm \exp(-A(T-t)).$$

We see that if $A > 0$, then the solution $\varphi(t)$ decays as t decreases from T, and the case $A > 0$ is thus the "stable case". If $A < 0$ then the exponential factor $\exp(-AT)$ enters, and depending on the size of A this case is "unstable". More precisely, we conclude directly from the explicit solution formula that

Theorem 81.2 *The stability factors $S_c(T)$ and $S_d(T)$ satisfy if $A < 0$:*

$$S_d(T) \le \exp(|A|T), \quad S_c(T) \le \exp(|A|T), \tag{81.11}$$

if $A \ge 0$:

$$S_d(T) \le 1, \quad S_c(T) \le 1 \tag{81.12}$$

if $A = i$:

$$S_d(T) \le 1, \quad S_c(T) \le T, \tag{81.13}$$

if $A = \sin(t)$:

$$S_d(T) \le \exp(1), \quad S_c(T) \le exp(1)T. \tag{81.14}$$

Proof: Changing variables $T - t \to t$, we can write the dual equation as the forward-in-time problem $\dot\varphi = -A\varphi$ for $t > 0$, $\varphi(0) = 1$ with solution $\exp(-At)$, if A is constant. We note that if $A > 0$, then

$$\int_0^T |\dot\varphi(t)|\,dt = \int_0^T A\exp(-At)\,dt = -\int_0^T \frac{d}{dt}\exp(-At)\,dt = 1-\exp(-AT) < 1. \tag{81.15}$$

Further, if $A < 0$, then

$$\int_0^T |\dot\varphi(t)|\,dt = \int_0^T \frac{d}{dt}\exp(-At)\,dt = \exp(-AT) - 1 \approx \exp(|A|T) \tag{81.16}$$

If $A = i$, then

$$\int_0^T |\dot\varphi(t)|\,dt = \int_0^T |\frac{d}{dt}\exp(-it)|\,dt = \int_0^T 1\,dt = T. \tag{81.17}$$

Finally, if $A = \sin(t)$ then $\varphi(t) = \exp(\cos(t))$, and so

$$|\varphi(T)| \leq \exp(1), \quad \int_0^T |\dot{\varphi}(t)|\,dt \leq \exp(1)T. \tag{81.18}$$

■.

The size of the stability factors indicate the degree of stability of the solution $u(t)$ being computed. If the stability factors are large, the residuals $R(U(t))$ and $e(0)$ have to be made correspondingly smaller by choosing smaller time steps and the computational problem is more demanding.

81.3 A Priori Error Estimate

The a posteriori error estimate (81.19) estimates the error in terms of the computed solution $U(t)$. There is a corresponding *a priori error estimate* with $R(U)$ replaced by $R(u_k)$ where u_k is the piecewise linear interpolant of the exact solution $u(t)$ taking on the same values at the discrete time levels nk. In this case the stability factors measure the stability of a corresponding *discrete dual problem*.

How big is then $R(u_k)$? Well, with piecewise linear interpolation, we have $|\dot{u} - \dot{u}_k| \approx k|\ddot{u}|$, and thus the a priori estimate takes the form

$$|u(T) - U(T)| \leq S_c(T)C(u)k^2 + S_d(T)|e(0)|, \tag{81.19}$$

where $C(u) = max_t|\ddot{u}(t)|$. In short, Midpoint Euler is *second-order accurate* with error proportional to k^2. Backward Euler and Forward Euler are *first order accurate* with error proportional to k.

81.4 Generalization

The above error analysis extends to a general IVP $\dot{u}(t) = f(u(t))$ for $t > 0$, $u(0) = u^0$, as shown in Chapter (159).

81.5 To Think About

- Show that the a posteriori estimate (81.19) directly extends to variable time steps k_n with k_nR_n replacing kR_n.
- For a basic aspect of duality in error estimation, see Error Control by Duality.

93
Solving Linear Algebraic Systems

All thought is a kind of computation. (Hobbes)

93.1 Introduction

We are interested in solving a system of linear equations

$$Ax = b,$$

where A is a given $n \times n$ matrix and $b \in \mathbb{R}^n$ is a given n-vector and we seek the solution vector $x \in \mathbb{R}^n$. We recall that if A is non-singular with non-zero determinant, then the solution $x \in \mathbb{R}^n$ is theoretically given by Cramer's formula. However if n is large, the computational work in using Cramer's formula is prohibitively large, so we need to find a more efficient means of computing the solution.

We shall consider two types of methods for solving the system $Ax = b$: (i) *direct methods* based on *Gaussian elimination* that theoretically produce a solution after a finite number of arithmetic operations, and (ii) *iterative methods* that produce a generally infinite sequence of increasingly accurate approximations.

93.2 Direct Methods

We begin by noting that some linear systems are easier to solve than others. For example if $A = (a_{ij})$ is *diagonal*, which means that $a_{ij} = 0$ if $i \neq j$,

then the system is solved in n operations: $x_i = b_i/a_{ii}$, $i = 1, ..., n$. Further, if the matrix is *upper triangular*, which means that $a_{ij} = 0$ if $i > j$, or *lower triangular*, which means that $a_{ij} = 0$ if $i < j$, then the system can be solved by *backward substitution* or *forward substitution* respectively; see Fig. 93.1 for an illustration of these different types. For example if A is

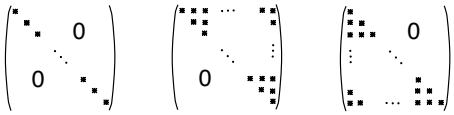

FIGURE 93.1. The pattern of entries in diagonal, upper, and lower triangular matrices. A "$*$" denotes a possibly nonzero entry.

upper triangular, the "pseudo-code" shown in Fig. 93.2 solves the system $Ax = b$ for the vector $x = (x_i)$ given the vector $b = (b_i)$ (assuming that $a_{kk} \neq 0$): In all three cases, the systems have a unique solution as long as

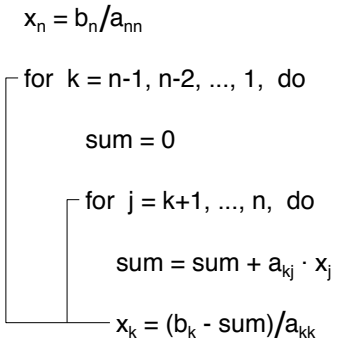

FIGURE 93.2. An algorithm for solving an upper triangular system by back substitution.

the diagonal entries of A are nonzero.

Direct methods are based on Gaussian elimination, which in turn is based on the observation that the solution of a linear system is not changed under the following *elementary row operations*:

- interchanging two equations
- adding a multiple of one equation to another
- multiplying an equation by a nonzero constant.

The idea behind Gaussian elimination is to transform using these operations a given system into an upper triangular system, which is solved by

back substitution. For example, to solve the system

$$\begin{aligned} x_1 + x_2 + x_3 &= 1 \\ x_2 + 2x_3 &= 1 \\ 2x_1 + x_2 + 3x_3 &= 1, \end{aligned}$$

we first subtract 2 times the first equation from the third to get the equivalent system,

$$\begin{aligned} x_1 + x_2 + x_3 &= 1 \\ x_2 + 2x_3 &= 1 \\ -x_2 + x_3 &= -1. \end{aligned}$$

We define the *multiplier* to be the factor 2. Next, we subtract -1 times the second row from the third to get

$$\begin{aligned} x_1 + x_2 + x_3 &= 1 \\ x_2 + 2x_3 &= 1 \\ 3x_3 &= 0. \end{aligned}$$

In this case, the multiplier is -1. The system is now upper triangular and using back substitution, we obtain $x_3 = 0$, $x_2 = 1$, and $x_1 = 0$. Gaussian elimination can be coded in a straightforward way using matrix notation.

Matrix Factorization

There is another way to view Gaussian elimination that is useful for the purposes of programming and handling special cases. Namely, Gaussian elimination is equivalent to computing a *factorization* of the coefficient matrix, $A = LU$, where L is a lower triangular and U an upper triangular $n \times n$ matrix. Given such a factorization of A, solving the system $Ax = b$ is straightforward. We first set $y = Ux$, then solve $Ly = b$ by forward substitution and finally solve $Ux = y$ by backward substitution.

To see that Gaussian elimination gives an LU factorization of A, consider the example above. We performed row operations that brought the system into upper triangular form. If we view these operations as row operations on the matrix A, we get the sequence

$$\begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 2 \\ 2 & 1 & 3 \end{pmatrix} \rightarrow \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 2 \\ 0 & -1 & 1 \end{pmatrix} \rightarrow \begin{pmatrix} 1 & 1 & 2 \\ 0 & 1 & 2 \\ 0 & 0 & 3 \end{pmatrix},$$

which is an upper triangular matrix. This is the "U" in the LU decomposition.

The matrix L is determined by the observation that the row operations can be performed by multiplying A on the left by a sequence of special

matrices called *Gauss transformations*. These are lower triangular matrices that have at most one nonzero entry in the off-diagonal positions and 1s down the diagonal. We show a Gauss transformation in Fig. 93.3. Multiplying A on the left by the matrix in Fig. 93.3 has the effect of adding

$$\begin{pmatrix}
1 & 0 & & & & \cdots & & & 0 \\
0 & 1 & 0 & & & & & & 0 \\
 & \ddots & 1 & \ddots & & 0 & & & \\
\vdots & & & & & & & & \vdots \\
 & 0 & 0 & 0 & \ddots & 0 & & & \\
 & 0 & \alpha_{ij} & 0 & \ddots & 1 & \ddots & & \\
 & 0 & 0 & 0 & & & \ddots & & \\
 & & & & & & 0 & 1 & 0 \\
0 & & \cdots & & & & & 0 & 1
\end{pmatrix}$$

FIGURE 93.3. A Gauss transformation.

α_{ij} times row j of A to row i of A. Note that the inverse of this matrix is obtained changing α_{ij} to $-\alpha_{ij}$; we will use this below.

To perform the first row operation on A above, we multiply A on the left by

$$L_1 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -2 & 0 & 1 \end{pmatrix},$$

to get

$$L_1 A = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 2 \\ 0 & -1 & -1 \end{pmatrix}.$$

The effect of pre-multiplication by L_1 is to add $-2\times$ row 1 of A to row 3. Note that L_1 is lower triangular and has ones on the diagonal.

Next we multiply $L_1 A$ on the left by

$$L_2 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix},$$

and get

$$L_2 L_1 A = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 2 \\ 0 & 0 & 3 \end{pmatrix} = U.$$

L_2 is also lower triangular with ones on the diagonal. It follows that $A = L_1^{-1}L_2^{-1}U$ or $A = LU$, where

$$L = L_1^{-1}L_2^{-1} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 2 & -1 & 1 \end{pmatrix}.$$

It is easy to see that L is also lower triangular with 1's on the diagonal with the multipliers (with sign change) occurring at the corresponding positions. We thus get the factorization

$$A = LU = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 2 & -1 & 1 \end{pmatrix} \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 2 \\ 0 & 0 & 3 \end{pmatrix}.$$

Note that the entries in L below the diagonal are exactly the multipliers used to perform Gaussian elimination on A.

A general linear system can be solved in exactly the same fashion by Gaussian elimination using a sequence of Gauss transformations to obtain a factorization $A = LU$.

An LU factorization can be performed *in situ* using the storage space allotted to the matrix A. The fragment of code shown in Fig. 93.4 computes the LU factorization of A, storing U in the upper triangular part of A and storing the entries in L below the diagonal in the part of A below the diagonal. We illustrate the storage of L and U in Fig. 93.5.

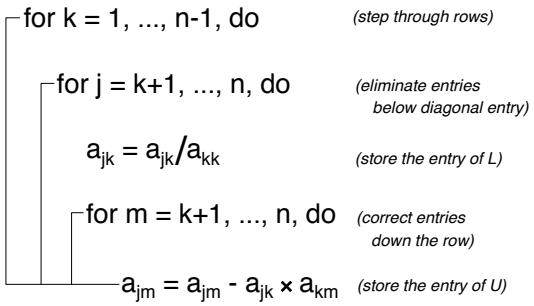

FIGURE 93.4. An algorithm to compute the LU factorization of A that stores the entries of L and U in the storage space of A.

Measuring the Cost

The cost of solving a linear system using a direct method is measured in terms of computer time. In practice, the amount of computer time is

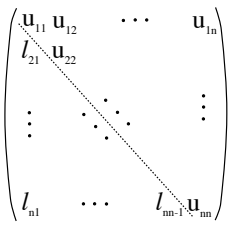

FIGURE 93.5. The matrix output from the algorithm in Fig. 93.4. L and U are stored in the space allotted to A.

proportional to the number of arithmetic and storage operations the computer uses to compute the solution. It is traditional (on a sequential computer) to simplify the cost calculation by equating storing a value, addition, and subtraction and equating multiplication and division when counting operations. Moreover, since multiplication (i.e. multiplications and divisions) generally cost much more than addition on older computers, it is also common to simply count the number of multiplications (=multiplications+divisions).

By this measure, the cost of computing the LU decomposition of an $n \times n$ matrix is $n^3 - n/3 = O(n^3/3)$. We introduce some new notation here, the big "O". The actual count is $n^3/3 - n/3$, however when n is large, the lower order term $-n/3$ becomes less significant. In fact,

$$\lim_{n\to\infty} \frac{n^3/3 - n/3}{n^3/3} = 1, \tag{93.1}$$

and this is the definition of the big "O". (Sometimes the big "O" notation means that the limit of the ratio of the two relevant quantities is any constant). With this notation, the operations count of the LU decomposition is just $O(n^3)$.

The cost of the forward and backward substitutions is much smaller:

Pivoting

During Gaussian elimination, it sometimes happens that the coefficient of a variable in the "diagonal position" becomes zero as a result of previous eliminations. When this happens of course, it is not possible to use that equation to eliminate the corresponding entries in the same column lying below the diagonal position. If the matrix is invertible, it is possible to find a non-zero coefficient in the same column and below the diagonal position, and by switching the two rows, the Gaussian elimination can proceed. This is called *zero pivoting*, or just *pivoting*.

Adding pivoting to the LU decomposition algorithm is straightforward. Before beginning the elimination using the current diagonal entry, we check

to see if that entry is non-zero. If it is zero, we search the entries below in the same column for the first non-zero value, then interchange the row corresponding to that non-zero entry with the row corresponding to the current diagonal entry which is zero. Because the row interchanges involve rows in the "un-factored" part of A, the form of L and U are not affected. We illustrate this in Fig. 93.6.

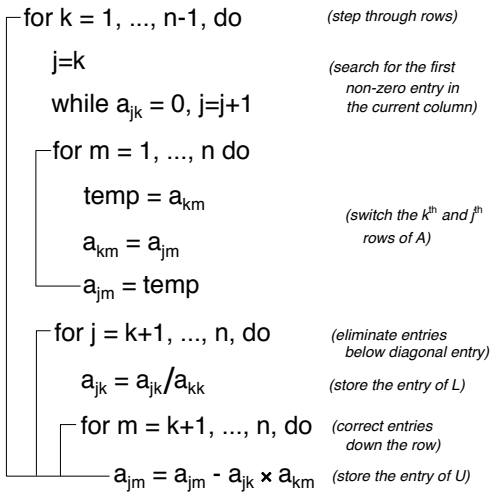

FIGURE 93.6. An algorithm to compute the LU factorization of A that used pivoting to avoid zero-valued diagonal entries.

To obtain the correct solution of the linear system $Ax = b$, we have to mirror all pivots performed on A in the data b. This is easy to do with the following trick. We define the vector of integers $p = (1\ 2\ \ldots\ n)^\top$. This vector is passed to the LU factorization routine and whenever two rows of A are interchanged, we interchange the corresponding entries in p. After getting the altered p vector back, we pass it to the forward/backward routine. Here, we address the vector b indirectly using the vector p, i.e., we use the vector with entries $(b_{p_i})_{i=1}^n$, which has the effect of interchanging the rows in b in the correct fashion.

There are additional reasons to pivot in practice. As we have noted, the computation of the LU decomposition can be sensitive to errors originating from the finite precision of the computer if the matrix A is close to being non-invertible. We discuss this further below. We mention here however that a special kind of pivoting, called *partial pivoting* can be used to reduce this sensitivity. The strategy behind partial pivoting is to search the entries in the same column and below the current diagonal entry for the largest in absolute value. The row corresponding to the largest entry in magnitude is interchanged with the row corresponding to the current entry at the diago-

nal. The use of partial pivoting generally gives more accurate results than factorization without partial pivoting. One reason is that partial pivoting insures that the multipliers in the elimination process are kept as small as possible and consequently the errors in each entry are magnified by as little as possible during the course of the Gaussian elimination. We illustrate this with an example. Suppose that we solve

$$\begin{aligned} .000100x_1 + 1.00x_2 &= 1.00 \\ 1.00x_1 + 1.00x_2 &= 2.00 \end{aligned}$$

on a computer that holds three digits. Without pivoting, we get

$$\begin{aligned} .000100x_1 + 1.00x_2 &= 1.00 \\ -10000x_2 &= -10000 \end{aligned}$$

which implies that $x_2 = 1$ and $x_1 = 0$. Note the large multiplier that is required for the elimination. Since the true answer is $x_1 = 1.0001$ and $x_2 = .9999$, the computed result has an error of 100% in x_1. If we switch the two rows before eliminating, which corresponds exactly to the partial pivoting strategy, we get

$$\begin{aligned} 1.00x_1 + 1.00x_2 &= 2.00 \\ 1.00x_2 &= 1.00 \end{aligned}$$

which gives $x_1 = x_2 = 1.00$ as a result.

93.3 Direct Methods for Special Systems

It is often the case that the matrices arising from the Galerkin finite element method applied to a differential equation have special properties that can be useful during the solution of the associated algebraic equations. For example, the stiffness matrix for the Galerkin finite element approximation of the two-point boundary value problem with no convection is symmetric, positive-definite, and tridiagonal. In this section, we examine a couple of different classes of problems that occur frequently.

Symmetric, Positive-Definite Systems

As we mentioned, symmetric, positive-definite matrices are often encountered when discretizing differential equations (especially if the spatial part of the differential equation is of the type called elliptic). If A is symmetric and positive-definite, then it can be factored as $A = BB^\top$ where B is a lower triangular matrix with positive diagonal entries. This factorization can be computed from the LU decomposition of A, but there is a *compact*

method of factoring A that requires only $O(n^3/6)$ multiplications called *Cholesky's method.*:

$$
\begin{aligned}
&b_{11} = \sqrt{a_{11}} \\
&b_{i1} = \frac{a_{i1}}{b_{11}}, \quad 2 \le i \le n, \\
&\begin{cases} b_{jj} = \left(a_{jj} - \sum_{k=1}^{j-1} b_{jk}^2\right)^{1/2}, \\ b_{ij} = \left(a_{ij} - \sum_{k=1}^{j-1} b_{ik} b_{jk}\right) / b_{jj}, \end{cases} \qquad 2 \le j \le n,\ j+1 \le i \le n
\end{aligned}
$$

This is called a compact method because it is derived by assuming that the factorization exists and then computing the coefficients of B directly from the equations obtained by matching coefficients in $BB^\top = A$. For example, if we compute the coefficient in the first row and column of $BB^\top$ we get b_{11}^2, which therefore must equal a_{11}. It is possible to do this because A is positive-definite and symmetric, which implies among other things that the diagonal entries of A remain positive throughout the factorization process and pivoting is not required when computing an LU decomposition.

Alternatively, the square roots in this formula can be avoided by computing a factorization $A = CDC^\top$ where C is a lower triangular matrix with ones on the diagonal and D is a diagonal matrix with positive diagonal coefficients.

Banded Systems

Banded systems are matrices with non-zero coefficients only in some number of diagonals centered around the main diagonal. In other words, $a_{ij} = 0$ for $j \le i - d_l$ and $j \ge i + d_u$, $1 \le i, j \le n$, where d_l is the *lower bandwidth*, d_u is the *upper bandwidth*, and $d = d_u + d_l - 1$ is called the *bandwidth*. We illustrate this in Fig. 93.7. The stiffness matrix computed for the two-point boundary value problem with no convection is an example of a tridiagonal matrix, which is a matrix with lower bandwidth 2, upper bandwidth 2, and bandwidth 3.

When performing the Gaussian elimination used to compute the LU decomposition, we see that the entries of A that are already zero do not have to be reduced further. If there are only relatively few diagonals with non-zero entries, then the potential saving is great. Moreover, there is no need to store the zero-valued entries of A. It is straightforward to adapt the LU factorization and forward/backward substitution routines to a banded pattern, once a storage scheme has been devised. For example, we can store

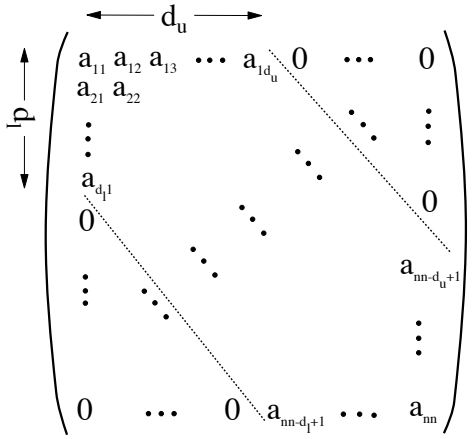

FIGURE 93.7. The notation for a banded matrix.

a tridiagonal matrix as a $3 \times n$ matrix:

$$\begin{pmatrix} a_{21} & a_{31} & 0 & & \cdots & & 0 \\ a_{12} & a_{22} & a_{32} & 0 & \cdots & & 0 \\ 0 & a_{13} & a_{23} & a_{33} & 0 & \cdots & \vdots \\ & \ddots & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & & & 0 & a_{1n-1} & a_{2n-1} & a_{3n-1} \\ 0 & & \cdots & & 0 & a_{1n} & a_{2n} \end{pmatrix}.$$

The routine displayed in Fig. 93.8 computes the LU factorization, while the routine in Fig. 93.9 performs the forward/backward substitution.

```
for k = 2, ..., n, do
    a_1k = a_1k / a_2k-1
    a_2k = a_2k - a_1k × a_3k-1
```

FIGURE 93.8. A routine for computing the LU factorization of a tridiagonal system.

The cost of this routine grows linearly with the dimension, rather than at a cubic rate as in the full case. Moreover, we use only the equivalent of six vectors of dimension n for storage. A more efficient version, derived as a compact method, uses even less.

This algorithm assumes that no pivoting is required to factor A. Pivoting during the factorization of a banded matrix raises the difficulty that the

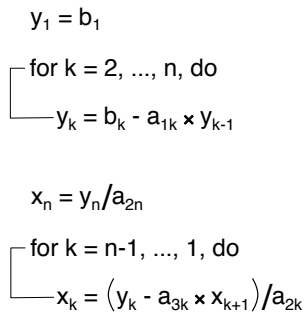

FIGURE 93.9. Using forward and backward substitution to solve a tridiagonal system given the LU factorization.

bandwidth becomes larger. This is easy to see in a tridiagonal matrix, in which case we have to store an extra vector to hold the extra elements above the diagonal that result if two adjacent rows are switched.

As for a tridiagonal matrix, it is straightforward to program special LU factorization and forward/backward substitution routines for a matrix with bandwidth d. The operations count is $O(nd^2/2)$ and the storage requirement is a matrix of dimension $d \times n$ if no pivoting is required. If d is much less than n, the savings in a special approach are considerable.

While it is true that if A is banded, then L and U are also banded, it is also true that in general L and U have non-zero entries in positions where A is zero. This is called *fill-in*. In particular, the stiffness matrix for a boundary value problem in several variables is banded and moreover most of the sub-diagonals in the band have zero coefficients. However, L and U do not have this property and we may as well treat A as if all the diagonals in the band have non-zero entries.

Banded matrices are one example of the class of *sparse* matrices. Recall that a sparse matrix is a matrix with mostly zero entries. As for banded matrices, it is possible to take advantage of sparsity to reduce the cost of factoring A in terms of time and storage. However, it is more difficult to do this than for banded matrices if the sparsity pattern puts non-zero entries at any location in the matrix. One approach to this problem is based on rearranging the equations and variables, or equivalently rearranging the rows and columns to form a banded system.

93.4 Iterative Methods

Instead of solving $Ax = b$ directly, we now consider iterative solution methods based on computing a sequence of approximations $x^{(k)}$, $k = 1, 2, ...$, such that

$$\lim_{k\to\infty} x^{(k)} = x \quad \text{or} \quad \lim_{k\to\infty} \|x^{(k)} - x\| = 0,$$

for some norm $\|\cdot\|$.

Note that the finite precision of a computer has a different effect on an iterative method than it has on a direct method. A theoretically convergent sequence can not reach its limit in general on a computer using a finite number of digits. In fact, at the point at which the change from one iterate to the next occurs outside the range of digits held by the computer, the sequence simply stops changing. Practically speaking, there is no point computing iterations past this point, even if the limit has not been reached. On the other hand, it is often sufficient to have less accuracy than the limit of machine precision, and thus it is important to be able to estimate the accuracy of the current iterate.

Minimization Algorithms

We first construct iterative methods for a linear system $Ax = b$ where A is symmetric and positive-definite. In this case, the solution x can be characterized equivalently as the solution of the quadratic minimization problem: find $x \in \mathbb{R}^n$ such that

$$F(x) \leq F(y) \quad \text{for all } y \in R^n, \tag{93.2}$$

where

$$F(y) = \frac{1}{2}(Ay, y) - (b, y),$$

with $(\cdot, \cdot)$ denoting the usual Euclidean scalar product.

We construct an iterative method for the solution of the minimization problem (93.2) based on the following simple idea: given an approximation $x^{(k)}$, compute a new approximation $x^{(k+1)}$ such that $F(x^{(k+1)}) < F(x^{(k)})$. On one hand, since F is a quadratic function, there must be a "downhill" direction from the current position, unless we are at the minimum. On the other hand, we hope that computing the iterates so that their function values are strictly decreasing, will force the sequence to converge to the minimum point x. Such an iterative method is called a *minimization method*.

Writing $x^{(k+1)} = x^{(k)} + \alpha_k d^{(k)}$, where $d^{(k)}$ is a *search direction* and α_k is a *step length*, by direct computation we get

$$F\big(x^{(k+1)}\big) = F\big(x^{(k)}\big) + \alpha_k \big(Ax^{(k)} - b, d^{(k)}\big) + \frac{\alpha_k^2}{2}\big(Ad^{(k)}, d^{(k)}\big),$$

where we used the symmetry of A to write $(Ax^{(k)}, d^{(k)}) = (x^{(k)}, Ad^{(k)})$. If the step length is so small that the second order term in α_k can be neglected, then the direction $d^{(k)}$ in which F decreases most rapidly, or the direction of *steepest descent*, is

$$d^{(k)} = -(Ax^{(k)} - b) = -r^{(k)},$$

which is the opposite direction to the residual error $r^{(k)} = Ax^{(k)} - b$. This suggests using an iterative method of the form

$$x^{(k+1)} = x^{(k)} - \alpha_k r^{(k)}. \tag{93.3}$$

A minimization method with this choice of search direction is called a *steepest descent method*. The direction of steepest descent is perpendicular to the *level curve* of F through $x^{(k)}$, which is the curve in the graph of F generated by the points where F has the same value as at $x^{(k)}$. We illustrate this in Fig. 93.10.

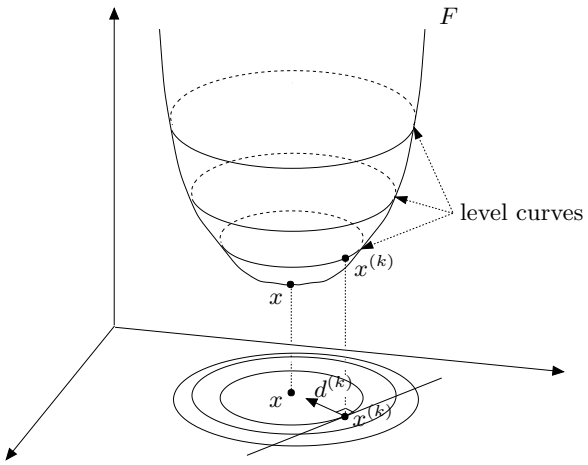

FIGURE 93.10. The direction of steepest descent of F at a point is perpendicular to the level curve of F through the point.

It remains to choose the step lengths α_k. Staying with the underlying principle, we choose α_k to give the minimum value of F in the direction of $d^{(k)}$ starting from $x^{(k)}$. Differentiating $F(x^{(k)} + \alpha_k r^{(k)})$ with respect to α_k and setting the derivative zero gives

$$\alpha_k = -\frac{\big(r^{(k)}, d^{(k)}\big)}{\big(d^{(k)}, Ad^{(k)}\big)}. \tag{93.4}$$

As a simple illustration, we consider the case

$$A = \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix}, \quad 0 < \lambda_1 < \lambda_2, \tag{93.5}$$

and $b = 0$, corresponding to the minimization problem

$$\min_{y \in \mathbb{R}^n} \frac{1}{2}\big(\lambda_1 y_1^2 + \lambda_2 y_2^2\big),$$

with solution $x = 0$.

Applying (93.3) to this problem, we iterate according to

$$x^{(k+1)} = x^{(k)} - \alpha_k A x^{(k)},$$

using for simplicity a constant step length with $\alpha_k = \alpha$ instead of (93.4). In Fig. 93.11, we plot the iterations computed with $\lambda_1 = 1$, $\lambda_2 = 9$, and $x^{(0)} = (9, 1)^\top$. The convergence in this case is quite slow. The reason is that if $\lambda_2 \gg \lambda_1$, then the search direction $-(\lambda_1 x_1^{(k)}, \lambda_2 x_2^{(k)})^\top$ and the direction $-(x_1^{(k)}, x_2^{(k)})^\top$ to the solution at the origin, are very different. As a result the iterates swing back and forth across the long, narrow "valley".

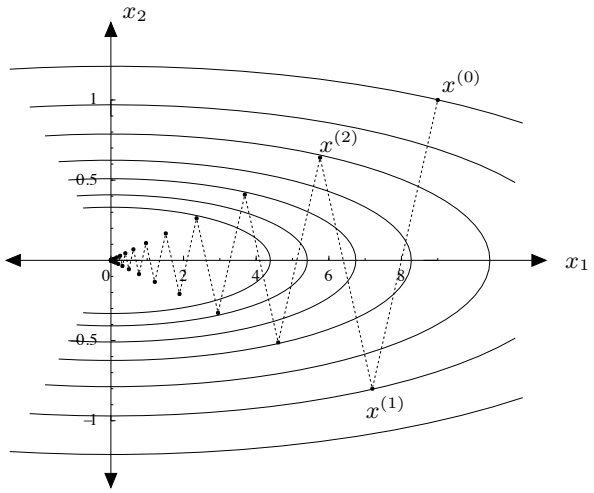

FIGURE 93.11. A sequence generated by the steepest descent method for (93.5) plotted together with some level curves of F.

It turns out that the rate at which the steepest descent method converges in general depends on the *condition number* $\kappa(A) = \lambda_n/\lambda_1$ of A, where $\lambda_1 \leq \lambda_2 \leq ... \leq \lambda_n$ are the eigenvalues of A (counted with multiplicity). In other words, the condition number of a symmetric positive definite matrix is the ratio of the largest eigenvalue to the smallest eigenvalue.

The general definition of the condition number of a matrix A in terms of a norm $\|\cdot\|$ is $\kappa(A) = \|A\|\|A^{-1}\|$. In the $\|\cdot\|_2$ norm, the two definitions are equivalent for symmetric matrices. Using any definition, a matrix is said to be *ill-conditioned* if the $\log(\kappa(A))$ is of the order of the number of digits used in the computer. As we said, we can expect to have difficulty solving an ill-conditioned system; which in terms of direct methods means large errors due to rounding errors and in terms of iterative methods means slow convergence.

We now analyze the steepest descent method for $Ax = b$ in the case of a constant step length α, where we iterate according to

$$x^{(k+1)} = x^{(k+1)} - \alpha(Ax^{(k)} - b).$$

Since the exact solution x satisfies $x = x - \alpha(Ax - b)$, we get the following equation for the error $e^{(k)} = x - x^{(k)}$:

$$e^{(k+1)} = (I - \alpha A)e^{(k)}.$$

The iterative method converges if the error tend to zero. Taking norms, we get

$$\|e^{(k+1)}\| \leq \mu \|e^{(k)}\| \tag{93.6}$$

where we use the spectral estimate (92.16) to write

$$\mu = \|I - \alpha A\| = \max_j |1 - \alpha\lambda_j|,$$

since the eigenvalues of the matrix $I - \alpha A$ are $1 - \alpha\lambda_j$, $j = 1, ..., n$. Iterating this estimate we get

$$\|e^{(k+1)}\| \leq \mu^k \|e^{(0)}\|, \tag{93.7}$$

where $e^{(0)}$ is the initial error.

To understand when (93.6), or (93.7), guarantees convergence, consider the scalar sequence $\{\lambda^k\}$ for $k \geq 0$. If $|\lambda| < 1$, then $\lambda^k \to 0$; if $\lambda = 1$, then the sequence is always 1; if $\lambda = -1$, the sequence alternates between 1 and -1 and does not converge; and if $|\lambda| > 1$, then the sequence diverges. Therefore if we want the iteration to converge for any initial value, then we must choose α so that $\mu < 1$. Since the λ_j are positive by assumption, $1 - \alpha\lambda_j < 1$ automatically, and we can guarantee that $1 - \alpha\lambda_j > -1$ if α satisfies $\alpha < 2/\lambda_n$. Choosing $\alpha = 1/\lambda_n$, which is not so far from optimal, we get

$$\mu = 1 - 1/\kappa(A).$$

If $\kappa(A)$ is large, then the convergence can be slow because then the reduction factor $1 - 1/\kappa(A)$ is close to one. More precisely, the number of steps required to lower the error by a given amount is proportional to the condition number.

When an iteration converges in this fashion, i.e. the error decreases (more or less) by a given factor in each iteration, then we say that the iteration converges *linearly*. We define the *rate of convergence* to be $-\log(\mu)$. The motivation is that the number of iterations are required to reduce the error by a factor of 10^{-m} is approximately $-m\log(\mu)$. Note that a faster rate of convergence means a smaller value of μ.

This is an *a priori* estimate of the error reduction per iteration, since we estimate the error before the computation. Such an analysis must account for the slowest possible rate of convergence because it holds for all initial vectors.

Consider the system $Ax = 0$ with

$$A = \begin{pmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{pmatrix}, \tag{93.8}$$

where $0 < \lambda_1 < \lambda_2 < \lambda_3$. For an initial guess $x^{(0)}=(x_1^0, x_2^0, x_3^0)^\top$, the steepest descent method with $\alpha = 1/\lambda_3$ gives the sequence

$$x^{(k)} = \left(\big(1 - \frac{\lambda_1}{\lambda_3}\big)^k x_1^0, \big(1 - \frac{\lambda_2}{\lambda_3}\big)^k x_2^0, 0 \right), \quad k = 1, 2, ...,$$

and,

$$\|e^{(k)}\| = \sqrt{\left(1 - \frac{\lambda_1}{\lambda_3}\right)^{2k} \big(x_1^0\big)^2 + \left(1 - \frac{\lambda_2}{\lambda_3}\right)^{2k} \big(x_2^0\big)^2}, \quad k = 1, 2, ...$$

Thus for a general initial guess, the size of the error is given by the root mean square average of the corresponding iterate and the rate that the errors decrease is the root mean square average of the rates of decrease of the components. Therefore, depending on the initial vector, initially the iterates will generally converge more quickly than the rate of decrease of the first, i.e. slowest, component. In other words, more quickly than the rate predicted by (93.6), which bounds the rate of decrease of the errors by the rate of decrease in the slowest component. However, as the iteration proceeds, the second component eventually becomes much smaller than the first component (as long as $x_1^0 \neq 0$) and we can neglect that term in the expression for the error, i.e.

$$\|e^{(k)}\| \approx \big(1 - \frac{\lambda_1}{\lambda_3}\big)^k |x_1^0| \quad \text{for } k \text{ sufficiently large.} \tag{93.9}$$

In other words, the rate of convergence of the error for almost all initial vectors eventually becomes dominated by the rate of convergence of the slowest component. It is straightforward to show that the number of iterations that we have to wait for this approximation to be valid is determined by the relative sizes of the first and second components of $x^{(0)}$.

This simple error analysis does not apply to the unmodified steepest descent method with varying α_k. However, it is generally true that the rate of convergence depends on the condition number of A, with a larger condition number meaning slower convergence. If we again consider the 2×2 example (93.5) with $\lambda_1 = 1$ and $\lambda_2 = 9$, then the estimate (93.6) for the simplified method suggests that the error should decrease by a factor of $1 - \lambda_1/\lambda_2 \approx .89$ in each iteration. The sequence generated by $x^{(0)} = (9, 1)^\top$ decreases by exactly .8 in each iteration. The simplified analysis over-predicts the rate of convergence for this particular sequence,

though not by a lot. By way of comparison, if we choose $x^{(0)} = (1,1)^\top$, we find that the ratio of successive iterations alternates between $\approx .126$ and $\approx .628$, because α_k oscillates in value, and the sequence converges much more quickly than predicted. On the other hand, there are initial guesses leading to sequences that converge at the predicted rate.

The stiffness matrix A of a linear second order two-point boundary value problem with no convection is symmetric and positive-definite, and its condition number $\kappa(A) \propto h^{-2}$. Therefore the convergence of the steepest descent method is very slow if the number of mesh points is large.

A General Framework for Iterative Methods

We now briefly discuss iterative methods for a general, linear system $Ax = b$, following the classical presentation of iterative methods in Isaacson and Keller ([**?**]). Recall that some matrices, like diagonal and triangular matrices, are relatively easy and cheap to invert, and Gaussian elimination can be viewed as a method of factoring A into such matrices. One way to view an iterative method is an attempt to approximate A^{-1} by the inverse of a part of A that is easier to invert. This is called an approximate inverse of A, and we use this to produce an approximate solution to the linear system. Since we don't invert the matrix A, we try to improve the approximate solution by repeating the partial inversion over and over. With this viewpoint, we start by *splitting* A into two parts:

$$A = N - P,$$

where the part N is chosen so that the system $Ny = c$ for some given c is relatively inexpensive to solve. Noting that the true solution x satisfies $Nx = Px + b$, we compute $x^{(k+1)}$ from $x^{(k)}$ by solving

$$Nx^{(k+1)} = Px^{(k)} + b \quad \text{for } k = 1, 2, ..., \tag{93.10}$$

where $x^{(0)}$ is an initial guess. For example, we may choose N to be the diagonal of A:

$$N_{ij} = \begin{cases} a_{ij}, & i = j, \\ 0, & i \neq j, \end{cases}$$

or triangular:

$$N_{ij} = \begin{cases} a_{ij}, & i \geq j, \\ 0, & i < j. \end{cases}$$

In both cases, solving the system $Nx^{(k+1)} = Px^{(k)} + b$ is cheap compared to doing a complete Gaussian elimination on A. so we could afford to do it many times.

As an example, suppose that

$$A = \begin{pmatrix} 4 & 1 & 0 \\ 2 & 5 & 1 \\ -1 & 2 & 4 \end{pmatrix} \quad \text{and} \quad b = \begin{pmatrix} 1 \\ 0 \\ 3 \end{pmatrix}, \tag{93.11}$$

and we choose

$$N = \begin{pmatrix} 4 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 4 \end{pmatrix} \quad \text{and} \quad P = \begin{pmatrix} 0 & -1 & 0 \\ -2 & 0 & -1 \\ 1 & -2 & 0 \end{pmatrix},$$

in which case the equation $Nx^{(k+1)} = Px^{(k)} + b$ reads

$$\begin{aligned} 4x_1^{k+1} &= -x_2^k + 1 \\ 5x_2^{k+1} &= -2x_1^k - x_3^k \\ 4x_3^{k+1} &= x_1^k - 2x_2^k + 3. \end{aligned}$$

Being a diagonal system it is easily solved, and choosing an initial guess and computing, we get

$$x^{(0)} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix},\ x^{(1)} = \begin{pmatrix} 0 \\ -.6 \\ .5 \end{pmatrix},\ x^{(2)} = \begin{pmatrix} .4 \\ -.1 \\ 1.05 \end{pmatrix},\ x^{(3)} = \begin{pmatrix} .275 \\ -.37 \\ .9 \end{pmatrix},$$
$$x^{(4)} = \begin{pmatrix} .3425 \\ -.29 \\ 1.00375 \end{pmatrix},\ \cdots\ x^{(15)} = \begin{pmatrix} .333330098 \\ -.333330695 \\ .999992952 \end{pmatrix},\ \cdots$$

The iteration appears to converge to the true solution $(1/3, -1/3, 1)^\top$.

In general, we could choose $N = N_k$ and $P = P_k$ to vary with each iteration.

To analyze the convergence of (93.10), we subtract (93.10) from the equation $Nx = Px + b$ satisfied by the true solution to get an equation for the error $e^{(k)} = x - x^{(k)}$:

$$e^{(k+1)} = Me^{(k)},$$

where $M = N^{-1}P$ is the *iteration matrix*. Iterating on k gives

$$e^{(k+1)} = M^{k+1}e^{(0)}. \tag{93.12}$$

Rephrasing the question of convergence, we are interested in whether $e^{(k)} \to 0$ as $k \to \infty$. By analogy to the scalar case discussed above, if M is "small", then the errors $e^{(k)}$ should tend to zero. Note that the issue of convergence is independent of the data b.

If $e^{(0)}$ happens to be an eigenvector of M, then it follows from (93.12)

$$\|e^{(k+1)}\| = |\lambda|^{k+1}\|e^{(0)}\|,$$

and we conclude that if the method converges then we must have $|\lambda| < 1$ (or $\lambda = 1$). Conversely, one can show that if $|\lambda| < 1$ for all eigenvalues of M, then the method (93.10) indeed does converge:

Theorem 93.1 *An iterative method converges for all initial vectors if and only if every eigenvalue of the associated iteration matrix is less than one in magnitude.*

This theorem is often expressed using the *spectral radius* $\rho(M)$ of M, which is the maximum of the magnitudes of the eigenvalues of A. An iterative method converges for all initial vectors if and only if $\rho(M) < 1$. In general, the asymptotic limit of the ratio of successive errors computed in $\| \ \|_\infty$ is close to $\rho(M)$ as the number of iterations goes to infinity. We define the *rate of convergence* to be $R_M = -\log(\rho(M))$. The number of iterations required to reduce the error by a factor of 10^m is approximately m/R_M.

Practically speaking, "asymptotic" means that the ratio can vary as the iteration proceeds, especially in the beginning. In previous examples, we saw that this kind of a priori error result can underestimate the rate of convergence even in the special case when the matrix is symmetric and positive-definite (and therefore has an orthonormal basis of eigenvectors) and the iterative method uses the steepest descent direction. The general case now considered is more complicated, because interactions may occur in direction as well as magnitude, and a spectral radius estimate may overestimate the rate of convergence initially. As an example, consider the non-symmetric (even non-normal) matrix

$$A = \begin{pmatrix} 2 & -100 \\ 0 & 4 \end{pmatrix} \tag{93.13}$$

choosing

$$N = \begin{pmatrix} 10 & 0 \\ 0 & 10 \end{pmatrix} \text{ and } P = \begin{pmatrix} 8 & 100 \\ 0 & 6 \end{pmatrix} \text{ gives } M = \begin{pmatrix} .9 & 10 \\ 0 & .8 \end{pmatrix}.$$

In this case, $\rho(M) = .9$ and we expect the iteration to converge. Indeed it does converge, but the errors become quite large before they start to approach zero. We plot the iterations starting from $x^{(0)} = (1, 1)^\top$ in Fig. 93.12.

The goal is obviously to choose an iterative method so that the spectral radius of the iteration matrix is small. Unfortunately, computing $\rho(M)$ in the general case is much more expensive than solving the original linear system and is impractical in general. We recall that $|\lambda| \leq \|A\|$ holds for any norm and any eigenvalue λ of A. The following theorem indicates a practical way to check for convergence.

Theorem 93.2 *Assume that $\|N^{-1}P\| \leq \mu$ for some constant $\mu < 1$ and matrix norm $\| \cdot \|$. Then the iteration converges and $\|e^{(k)}\| \leq \mu^k \|e^{(0)}\|$ for $k \geq 0$.*

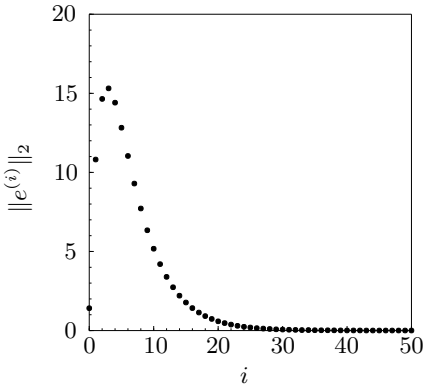

FIGURE 93.12. The results of an iterative method computed using a non-normal matrix.

This theorem is also an a priori convergence result and suffers from the same deficiency as the analysis of the simplified steepest descent method presented above. In fact, choosing an easily computable matrix norm, like $\| \ \|_\infty$, generally leads to an even more inaccurate estimate of the convergence rate than would be obtained by using the spectral radius. In the worst case, it is entirely possible that $\rho(M) < 1 < \|M\|$ for the chosen norm, and hence the iterative method converges even though the theorem does not apply. The amount of "slack" in the bound in Theorem 93.2 depends on how much larger $\|A\|_\infty$ is than $\rho(A)$.

For the 3×3 example (93.11), we compute $\|N^{-1}P\|_\infty = 3/4 = \lambda$ and therefore we know the sequence converges. The theorem predicts that the error will get reduced by a factor of $3/4$ every iteration. If we examine the error of each iterate along with the ratios of successive errors after the first iteration:

i	$\|e^{(i)}\|_\infty$	$\|e^{(i)}\|_\infty / \|e^{(i-1)}\|_\infty$
0	1.333	
1	.5	.375
2	.233	.467
3	.1	.429
4	.0433	.433
5	.0194	.447
6	.00821	.424
7	.00383	.466
8	.00159	.414
9	.000772	.487

we find that after the first few iterations, the errors get reduced by a factor in the range of .4–.5 each iteration and not the factor $3/4$ predicted above. The ratio of $e^{(40)}/e^{(39)}$ is approximately .469. If we compute the eigenvalues of M, we find that $\rho(M) \approx .476$ which is close to the ratio of successive

errors. To decrease the initial error by a factor of 10^{-4} using the predicted decrease of .75 per iteration, we would compute 33 iterations, while only 13 iterations are actually needed.

We get different methods, and different rates of convergence, by choosing different N and P. The method used in the example above is called the *Jacobi* method. In general, this consists of choosing N to be the "diagonal part" of A and P to be the negative of the "off-diagonal" part of A. This gives the set of equations

$$x_i^{k+1} = -\frac{1}{a_{ii}}\left(\sum_{j\neq i} a_{ij}x_j^k - b_i\right), \quad i = 1, ..., n.$$

To derive a more sophisticated method, we write out these equations in Fig. 93.13. The idea behind the *Gauss-Seidel* method is to use the new

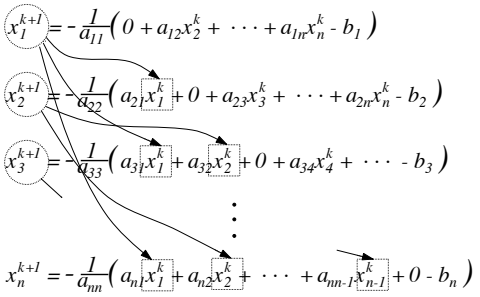

FIGURE 93.13. The Gauss-Seidel method substitutes new values of the iteration as they become available.

values of the approximation in these equations as they become known. The substitutions are drawn in Fig. 93.13. Presumably, the new values are more accurate than the old values, hence we might guess that this iteration will converge more quickly. The equations can be written

$$x_i^{k+1} = \frac{1}{a_{ii}}\left(-\sum_{j=1}^{i-1} a_{ij}x_j^{k+1} - \sum_{j=i+1}^{n} a_{ij}x_j^k + b_i\right).$$

If we decompose A into the sum of its lower triangular L, diagonal D, and upper triangular U parts, $A = L+D+U$, then the equations can be written $Dx^{(k+1)} = -Lx^{(k+1)} - Ux^{(k)} + b$ or

$$(D + L)x^{(k+1)} = -Ux^{(k)} + b.$$

Therefore, $N = D + L$ and $P = -U$. The iteration matrix is $M_{\mathrm{GS}} = N^{-1}P = -(D + L)^{-1}U$.

A diagonally dominant matrix often occurs when a parabolic problem is discretized. We have already seen the other case, if A is symmetric and positive-definite then the Gauss-Seidel method converges. This is quite hard to prove, see Isaacson and Keller ([**?**]) for details.

93.5 Estimating the Error of the Solution

The issue of estimating the error of the numerical solution of a linear system $Ax = b$ arises both in Gaussian elimination, because of the cumulative effects of round-off errors, and when using iterative methods, where we need a stopping criterion. Therefore it is important to be able to estimate the error in some norm with a fair degree of accuracy.

We discussed this problem in the context of iterative methods in the last section when we analyzed the convergence of iterative methods and Theorem 93.2 gives an *a priori* estimate for the convergence rate. It is an a priori estimate because the error is bounded before the computation begins. Unfortunately, as we saw, the estimate may not be very accurate on a particular computation, and it also requires the size of the initial error. In this section, we describe a technique of *a posteriori* error estimation that uses the approximation after it is computed to give an estimate of the error of that particular approximation.

We assume that x_c is a numerical solution of the system $Ax = b$ with exact solution x, and we want to estimate the error $\|x - x_c\|$ in some norm $\|\cdot\|$. We should point out that we are actually comparing the approximate solution $\tilde{x}_c$ of $\tilde{A}\tilde{x} = \tilde{b}$ to the true solution $\tilde{x}$, where $\tilde{A}$ and $\tilde{b}$ are the finite precision computer representations of the true A and b respectively. The best we can hope to do is compute $\tilde{x}$ accurately. To construct a complete picture, it would be necessary to examine the effects of small errors in A and b on the solution x. To simplify things, we ignore this part of the analysis and drop the $\tilde{}$. In a typical use of an iterative method, this turns out to be reasonable. It is apparently less reasonable in the analysis of a direct method, since the errors arising in direct methods are due to the finite precision. However, the initial error caused by storing A and b on a computer with a finite number of digits occurs only once, while the errors in the arithmetic operations involved in Gaussian elimination occur many times, so even in that case it is not an unreasonable simplification.

We start by considering the *residual error*

$$r = Ax_c - b,$$

which measures how well x_c solves the exact equation. Of course, the residual error of the exact solution x is zero but the residual error of x_c is not zero unless $x_c = x$ by some miracle. We now seek to estimate the unknown error $e = x - x_c$ in terms of the computable residual error r.

By subtracting $Ax - b = 0$ from $Ax_c - b = r$, we get an equation relating the error to the residual error:

$$Ae = -r. \tag{93.14}$$

This is an equation of the same from as the original equation and by solving it numerically by the same method used to compute x_c, we get an approximation of the error e. This simple idea will be used in a more sophisticated form below in the context of a posteriori error estimates for Galerkin methods.

We now illustrate this technique on the linear system arising in the Galerkin finite element discretization of a two-point boundary value problem with no convection. We generate a problem with a known solution so that we can compute the error and test the accuracy of the error estimate. We choose the true solution vector x with components $x_i = \sin(\pi ih)$, where $h = 1/(M+1)$, corresponding to the function $\sin(\pi x)$ and then compute the data by $b = Ax$, where A is the stiffness matrix. We use the Jacobi method, suitably modified to take advantage of the fact that A is tridiagonal, to solve the linear system. We use $\| \ \| = \| \ \|_2$ to measure the error.

We compute the Jacobi iteration until the residual error becomes smaller than a given *residual tolerance* RESTOL. In other words, we compute the residual $r^{(k)} = Ax^{(k)} - b$ after each iteration and stop the process when $\|r^{(k)}\| \leq$ RESTOL. We present computations using the stiffness matrix generated by a uniform discretization with $M = 50$ elements yielding a finite element approximation with an error of .0056 in the l_2 norm. We choose the value of RESTOL so that the error in the computation of the coefficients of the finite element approximation is about 1% of the error of the approximation itself. This is reasonable since computing the coefficients of the approximation more accurately would not significantly increase the overall accuracy of the approximation. After the computation of $x^{(k)}$ is complete, we use the Jacobi method to approximate the solution of (93.14) and compute the estimate of the error.

Using the initial vector $x^{(0)}$ with all entries equal to one, we compute 6063 Jacobi iterations to achieve $\|r\| <$ RESTOL $= .0005$. The actual error of $x^{(6063)}$, computed using the exact solution, is approximately .0000506233. We solve (93.14) using the Jacobi method for 6063 iterations, reporting the value of the error estimate every 400 iterations:

Iter.	Error Est.	Iter.	Error Est.	Iter.	Error Est.
1	0.00049862	2001	0.000060676	4001	0.000050849
401	0.00026027	2401	0.000055328	4401	0.000050729
801	0.00014873	2801	0.000052825	4801	0.000050673
1201	0.000096531	3201	0.000051653	5201	0.000050646
1601	0.000072106	3601	0.000051105	5601	0.000050634

We see that the error estimate is quite accurate after 6001 iterations and sufficiently accurate for most purposes after 2000 iterations. In general,

we do not require as much accuracy in the error estimate as we do in the solution of the system, so the estimation of the accuracy of the approximate solution is cheaper than the computation of the solution.

Since we estimate the error of the computed solution of the linear system, we can stop the Jacobi iteration once the error in the coefficients of the finite element approximation is sufficiently small so that we are sure the accuracy of the approximation will not be affected. This is a reasonable strategy given an estimate of the error. If we do not estimate the error, then the best strategy to guarantee that the approximation accuracy is not affected by the solution error is to compute the Jacobi iteration until the residual error is on the order of roughly 10^{-p}, where p is the number of digits that the computer uses. Certainly, there is not much point to computing further Jacobi iterations after this. If we assume that the computations are made in single precision, then $p \approx 8$. It takes a total of 11672 Jacobi iterations to achieve this level of residual error using the same initial guess as above. In fact, estimating the error and computing the coefficients of the approximation to a reasonable level of accuracy costs significantly less than this crude approach.

This approach can also be used to estimate the error of a solution computed by a direct method, provided the effects of finite precision are included. The added difficulty is that in general the residual error of a solution of a linear system computed with a direct method is small, even if the solution is inaccurate. Therefore, care has to be taken when computing the residual error because the possibility that subtractive cancellation makes the calculation of the residual error itself inaccurate. *Subtractive cancellation* is the name for the fact that the difference of two numbers that agree to the first i places has i leading zeroes. If only the first p digits of the numbers are accurate then their difference can have at most $p - i$ accurate significant digits. This can have severe consequences on the accuracy of the residual error if Ax_c and b agree to most of the digits used by the computer. One way to avoid this trouble is to compute the approximation in single precision and the residual in double precision (which means compute the product Ax_c in double precision, then subtract b). The actual solution of (93.14) is relatively cheap since the factorization of A has already been performed and only forward/backward substitution needs to be done.

93.6 The Conjugate Gradient Method

We learned above that solving an $n \times n$ linear system of equations $Ax = b$ with A symmetric positive definite using the gradient method, requires a number of iterations, which is proportional to the condition number $\kappa(A) = \lambda_n/\lambda_1$, where $\lambda_1 \leq ... \leq \lambda_n$ are the eigenvalues of A. Thus the number of

iteration will be large, maybe prohibitively so, if the condition number $\kappa(A)$ is large.

We shall now present a variant of the gradient method, referred as the *conjugate gradient method*, where the number of iterations scales instead like $\sqrt{\kappa(A)}$, which may be much smaller than $\kappa(A)$ if $\kappa(A)$ is large.

In the conjugate gradient method each new search direction is chosen to be orthogonal, with respect to the scalar product induced by the positive definite symmetric matrix A, which prevents choosing inefficient search directions as in the usual gradient method.

The conjugate gradient method may be formulated as follows: for $k = 1, 2, ...$ compute an approximate solution $x^k \in \mathbb{R}^n$ as the solution of the minimization problem

$$\min_{y \in K_k(A)} F(y) = \min_{y \in K_k(A)} \frac{1}{2}(Ay, y) - (b, y)$$

where $K_k(A)$ is the *Krylov space* spanned by the vectors $\{b, Ab, ..., A^{k-1}b\}$.

This is the same as defining x^k to be the projection of x onto $K_k(A)$ with respect to the scalar product $< y, z >$ on $\mathbb{R}^n \times \mathbb{R}^n$ defined by $< y, z >= (Ay, z)$, because we have using the symmetry of A and that $Ax = b$:

$$\frac{1}{2}(Ay, y) - (b, y) = \frac{1}{2} < y - x, y - x > - \frac{1}{2} < x, x > .$$

In particular, the conjugate gradient method has the following minimization property

$$\|x - x^k\|_A = \min_{y \in K_k(A)} \|x - y\|_A \leq \|p_k(A)x\|_A$$

where $p_k(x)$ is a polynomial of degree k with $p(0) = 1$, and $\| \cdot \|_A$ is the norm associated with the scalar product $< \cdot, \cdot >$, that is, $\|y\|_A^2 =< y, y >$. This follows by using that since $b = Ax$, we have that $K_k(A)$ is spanned by the vectors $\{Ax, A^2x, ..., A^kx\}$. In particular, we conclude that for all polynomials $p_k(x)$ of degree k such that $p_k(0) = 1$, we have

$$\|x - x^k\|_A \leq \max_{\lambda \in \Lambda} |p_k(\lambda)| \|x\|_A \tag{93.15}$$

where Λ is the set of eigenvalues of A. By choosing the polynomial $p_k(x)$ properly, e.g as a so-called *Chebyshev polynomial* $q_k(x)$ with the property that $q_k(x)$ is small on the interval $[\lambda_1, \lambda_n]$ containing the eigenvalues of A, one can prove that the number of iterations scales like $\sqrt{\kappa(A)}$ if n is large.

If n is not large, we have in particular from (93.15) that we get the exact solution after at most n iterations, since we may choose the polynomial $p_k(x)$ to be zero at the n eigenvalues of A.

We have now defined the conjugate gradient method through it structural properties: projection onto a Krylov space with respect to a certain

scalar product, and we now address the problem of actually computing the sequence x^k step by step. This is done as follows: For $k = 0, 1, 2, ...,$

$$x^{k+1} = x^k + \alpha_k d^k, \quad \alpha_k = -\frac{(r^k, d^k)}{< d^k, d^k >}, \tag{93.16}$$

$$d^{k+1} = -r^{k+1} + \beta_k d^k, \quad \beta_k = \frac{< r^{k+1}, d^k >}{< d^k, d^k >}, \tag{93.17}$$

where $r^k = Ax^k - b$ is the residual of the approximation x^k, and we choose $x^0 = 0$ and $d_0 = b$. Here, (93.17) signifies that the new search direction d^{k+1} gets new directional information from the new residual r^{k+1} and is chosen to be orthogonal (with respect to the scalar product $< \cdot, \cdot >$) to the old search direction d^k. Further, (93.16), expresses that x^{k+1} is chosen so as to to minimize $F(x^{(k)} + \alpha d^k)$ in α, corresponding to projection onto $K_{k+1}(A)$. We prove these properties in a sequence of problems below.

Note that if we choose the initial approximation x^0 different from zero, then we may reduce to the above case by considering instead the problem $Ay = b - Ax^0$ in y, where $y = x - x^0$.

93.7 GMRES

The conjugate gradient method for solving an $n \times n$ system $Ax = b$ builds on the matrix A being symmetric and positive definite. If A is non-symmetric or non-positive definite, but yet non-singular, then we may apply the conjugate gradient method to the least squares problem $A^\top Ax = A^\top b$, but the since the condition number of $A^\top A$ typically is the square of the condition number of A, the required number of iterations may be too large for efficiency.

Instead we may try the *Generalized Minimum Residual* method referred to as *GMRES*, which generates a sequence of approximations x^k of the solution x of $Ax = b$, satisfying for any polynomial $p_k(x)$ of degree at most k with $p_k(0) = 1$

$$\|Ax^k - b\| = \min_{y \in K_k(A)} \|Ay - b\| \leq \|p_k(A)b\|, \tag{93.18}$$

that is x^k is the element in the Krylov space $K_k(A)$ which minimizes the Euclidean norm of the residual $Ay - b$ with $y \in K_k(A)$. Assuming that the matrix A is *diagonalizable*, there exist a nonsingular matrix V so that $A = VDV^{-1}$, where D is a diagonal matrix with the eigenvalues of A on the diagonal. We then have that

$$\|Ax^k - b\| \leq \kappa(V) \max_{\lambda \in \Lambda} |p_k(\lambda)| \|b\|, \tag{93.19}$$

where Λ is the set of eigenvalues of A.

In the actual implementation of GMRES we use the *Arnoldi iteration*, a variant of the Gram-Schmidt orthogonalization, that constructs a sequence of matrices Q_k whose orthogonal column vectors span the successive Krylov spaces $K_k(A)$, and we write $x^k = Q_k c$ to get the following least squares problem:

$$\min_{c\in\mathbb{R}^k} \|AQ_n c - b\|. \tag{93.20}$$

The Arnoldi iteration is based on the identity $AQ_k = Q_{k+1}H_k$, where H_k is an *upper Hessenberg matrix* so that $h_{ij} = 0$ for all $i > j + 1$. Using this identity and multiplying from the left by Q_{k+1}^T gives us another equivalent least squares problem:

$$\min_{c\in\mathbb{R}^k} \|H_k c - Q_{k+1}^T b\|. \tag{93.21}$$

Recalling the construction of the Krylov spaces $K_k(A)$, in particular that $K_1(A)$ is spanned by b, we find that $Q_{k+1}^T b = \|b\|e_1$, where $e_1 = (1, 0, 0, ...)$, and we obtain the final form of the least squares problem to be solved in the GMRES iteration:

$$\min_{c\in\mathbb{R}^k} \|H_k c - \|b\|e_1\|. \tag{93.22}$$

This problem is now easy to solve due to the simpel structure of the Hessenberg matrix H_k.

In Figure 93.14 we compare the performance of the conjugate gradient method and GMRES for system with a tridiagonal 200×200 matrix with 1 on the diagonal, and random off-diagonal entries that take values in $(-0.5, 0.5)$ and the right hand side a random vector with values in $[-1, 1]$. The system matrix in this case is not symmetric, but it is strictly diagonally dominant and thus may be viewed asa perturbation of the identity matrix and should be easy to solve iteratively. We see that both the conjugate gradient method and GMRES converge quite rapidly, with GMRES winning in number of iterations.

In GMRES we need to store the basis vectors for the increasing Krylov space, which may be prohibitive for large systems requiring many iterations. To avoid this problem, we may restart GMRES when we have reached a maximal number of stored basis vector, by using as initial approximation x^0 the last approximation before restart. The trade-off is of course that a retarted GMRES may require more iterations for the same accuracy than GMRES without restart.

We now consider the more challenging problem of solving a 200×200 *stiffness matrix* system, that is a system with a tridiagonal matrix with 2 on the diagonal, and -1 on the off-diagonal (which is not strictly diagonally dominant). We will meet this type of system matrix in Chapter FEM for Two-Point Boundary Value Problems below, and we will see that it has a condition number proportional to the square of the number of unknowns. We thus expect the conjugate gradient method to require about

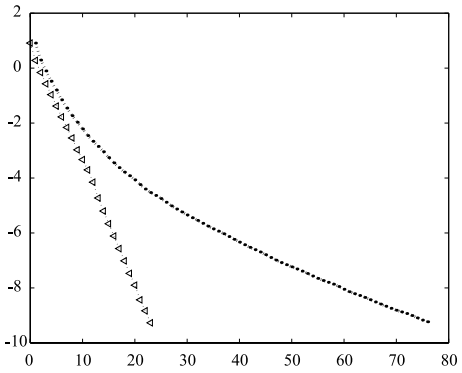

FIGURE 93.14. Log-plot of the residual versus the number of iterations for diagonal dominant random matrix, using the conjugate gradient method ($'\cdot'$) and GMRES ('triangles').

the same number of iterations as the number of unknowns. In Figure 93.15 we compare again the performance of the conjugate gradient method with the GMRES method, now restarted after 100 iterations. We find that the conjugate gradient method as expected converges quite slowly (and non monotonically), until immediate convergence at iteration 200 as predicted by theory. The GMRES iteration on the other hand has a monotone but still quite slow convergence in particular after each restart when the Krylov subspace is small.

In Figure 93.16 we compare different restart conditions for GMRES, and we find that there is a trade-off between the convergence rate and the memory consumption: few restarts give a faster convergence, but require more memory to store more basis vectors for the Krylov space. On the other hand we save memory by using more restarts, but then the convergence rate deteriorates.

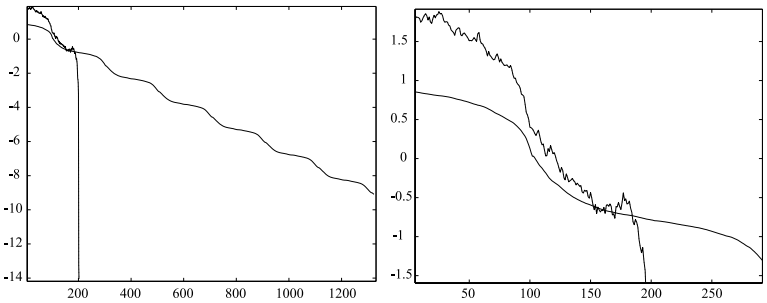

FIGURE 93.15. Log-plot of the residual versus the number of iterations for stiffness matrix, using the conjugate gradient method and GMRES, restarted after 100 iterations.

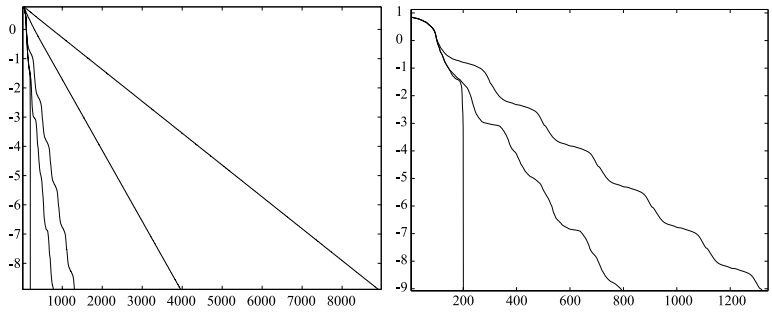

FIGURE 93.16. Log-plot of the residual versus the number of iterations for stiffness matrix using GMRES and restarted GMRES, restarted after 20,50,100,150 iterations (left), and a close-up on the cases of no restart and restart after 100 and 150 iterations (right).

Chapter 93 Problems

93.1. Using a similar format, write down algorithms to solve a diagonal system and then a lower triangular system using forward substitution. Determine the number of arithmetic operations needed to compute the solution.

93.2. Prove that multiplying a square matrix A on the left by the matrix in Fig. 93.3 has the effect of adding α_{ij} times row j of A to row i of A. Prove that the inverse of the matrix in Fig. 93.3 is obtained changing α_{ij} to $-\alpha_{ij}$

93.3. Show that the product of two Gauss transformations is a lower triangular matrix with ones on the diagonal and the inverse of a Gauss transformation is a Gauss transformation.

93.4. Solve the system

$$\begin{aligned} x_1 - x_2 - 3x_3 &= 3 \\ -x_1 + 2x_2 + 4x_3 &= -5 \\ x_1 + x_2 &= -2 \end{aligned}$$

by computing an LU factorization of the coefficient matrix and using forward/ backward substitution.

93.5. On some computers, dividing two numbers is up to ten times more expensive than computing the reciprocal of the denominator and multiplying the result with the numerator. Alter this code to avoid divisions. Note: the reciprocal of the diagonal element a_{kk} has to be computed just once.

93.6. Write some pseudo-code that uses the matrix generated by the code in Fig. 93.4 to solve the linear system $Ax = b$ using forward/backward substitution. Hint: the only missing entries of L are the 1s on the diagonal.

93.7. Show that the cost of a backward substitution using an upper triangular matrix of dimension $n \times n$ is $O(n^2/2)$.

93.8. Determine the cost of multiplying a $n \times n$ matrix with another.

93.9. One way to compute the inverse of a matrix is based on viewing the equation $AA^{-1} = I$ as a set of linear equations for the columns of A^{-1}. If $a^{(j)}$ denotes the j^{th} column of A^{-1}, then it satisfies the linear system

$$Aa^{(j)} = e_j$$

where e_j is the standard basis vector of $\mathbb{R}^n$ with a one in the j^{th} position. Use this idea to write a pseudo-code for computing the inverse of a matrix using LU factorization and forward/backward substitution. Note that it suffices to compute the LU factorization only once. Show that the cost of computing the inverse in this fashion is $O(4n^3/3)$.

93.10. Solve the system

$$\begin{aligned} x_1 + x_2 + x_3 &= 2 \\ x_1 + x_2 + 3x_3 &= 5 \\ -x_1 - 2x_3 &= -1. \end{aligned}$$

This requires pivoting.

93.11. Alter the LU decomposition and forward/backward routines to solve a linear system with pivoting.

93.12. Modify the code in Problem 93.11 to use partial pivoting.

93.13. Count the cost of Cholesky's method.

93.14. Compute the Cholesky factorization of

$$\begin{pmatrix} 4 & 2 & 1 \\ 2 & 3 & 0 \\ 1 & 0 & 2 \end{pmatrix}$$

93.15. Show that the operations count for solving a tridiagonal system using the solver described in Fig. 93.9 is $O(5n)$.

93.16. Find an algorithm to solve a tridiagonal system that stores only four vectors of dimension n.

93.17. A factorization of a tridiagonal solver can be derived as a compact method. Assume that A can be factored as

$$A = \begin{pmatrix} \alpha_1 & 0 & & \cdots & & 0 \\ \beta_2 & \alpha_2 & 0 & & & \vdots \\ 0 & \beta_3 & \alpha_3 & & & \\ \vdots & & & \ddots & 0 & \\ 0 & \cdots & 0 & \beta_n & \alpha_n & \end{pmatrix} \begin{pmatrix} 1 & \gamma_1 & 0 & \cdots & & 0 \\ 0 & 1 & \gamma_2 & 0 & & \\ \vdots & & \ddots & \ddots & & \\ & & & 1 & \gamma_{n-1} & \\ 0 & \cdots & & 0 & 1 & \end{pmatrix}$$

Multiply out the factors and equate the coefficients to get equations for α, β, and γ. Derive some code based on these formulas.

93.18. Write some code to solve the tridiagonal system resulting from the Galerkin finite element discretization of a two-point boundary value problem. Using 50 elements, compare the time it takes to solve the system with this tridiagonal solver to the time using a full LU decomposition routine.

93.19. Show that the operations count of a banded solver for a $n \times n$ matrix with bandwidth d is $O(nd^2/2)$.

93.20. Write code to solve a linear system with bandwidth five centered around the main diagonal. What is the operations count for your code?

93.21. Prove that the solution of (93.2) is also the solution of $Ax = b$.

93.22. Prove that the direction of steepest descent for a function F at a point is perpendicular to the level curve of F through the same point.

93.23. Prove (93.4).

93.24. Prove that the level curves of F in the case of (93.5) are ellipses with major and minor axes proportional to $1/\sqrt{\lambda_1}$ and $1/\sqrt{\lambda_2}$, respectively.

93.25. Compute the iteration corresponding to $\lambda_1 = 1$, $\lambda_2 = 2$, $\lambda_3 = 3$, and $x^{(0)} = (1, 1, 1)^\top$ for the system $Ax = 0$ with A defined in (93.8). Make a plot of the ratios of successive errors versus the iteration number. Do the ratios converge to the ratio predicted by the error analysis?

93.26. Prove that the estimate (93.9) generalizes to any symmetric positive-definite matrix A, diagonal or not. Hint: use the fact that there is a set of eigenvectors of A that form an orthonormal basis for $\mathbb{R}^n$ and write the initial vector in terms of this basis. Compute a formula for the iterates and then the error.

93.27. (a) Compute the steepest descent iterations for (93.5) corresponding to $x^{(0)} = (9, 1)^\top$ and $x^{(0)} = (1, 1)^\top$, and compare the rates of convergence. Try to make a plot like Fig. 93.11 for each. Try to explain the different rates of convergence.

(b) Find an initial guess which produces a sequence that decreases at the rate predicted by the simplified error analysis.

93.28. Prove that the method of steepest descent corresponds to choosing

$$N = N_k = \frac{1}{\alpha_k} I, \text{ and } P = P_k = \frac{1}{\alpha_k} I - A,$$

with suitable α_k in the general iterative solution algorithm.

93.29. Compute the eigenvalues and eigenvectors of the matrix A in (93.13) and show that A is not normal.

93.30. Prove that the matrix $\begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix}$ is normal.

93.31. Prove Theorem 93.2.

93.32. Compute 10 Jacobi iterations using the A and b in (93.11) and the initial guess $x^{(0)} = (-1,\ 1,\ -1)^\top$. Compute the errors and the ratios of successive errors and compare to the results above.

93.33. Repeat Problem 93.32 using

$$A = \begin{pmatrix} 4 & 1 & 100 \\ 2 & 5 & 1 \\ -1 & 2 & 4 \end{pmatrix} \quad \text{and} \quad b = \begin{pmatrix} 1 \\ 0 \\ 3 \end{pmatrix}.$$

Does Theorem 93.2 apply to this matrix?

93.34. Show that for the Jacobi iteration, $N = D$ and $P = -(L + U)$ and the iteration matrix is $M_{\mathrm{J}} = -D^{-1}(L + U)$

93.35. (a) Solve (93.11) using the Gauss-Seidel method and compare the convergence with that of the Jacobi method. Also compare $\rho(M)$ for the two methods. (b) Do the same for the system in Problem 93.33.

93.36. (Isaacson and Keller ([**?**])) Analyze the convergence of the Jacobi and Gauss-Seidel methods for the matrix

$$A = \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix}$$

in terms of the parameter ρ.

In general it is difficult to compare the convergence of the Jacobi method with that of the Gauss-Seidel method. There are matrices for which the Jacobi method converges and the Gauss-Seidel method fails and vice versa. There are two special classes of matrices for which convergence can be established without further computation. A matrix A is *diagonally dominant* if

$$|a_{ii}| > \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}|, \quad i = 1, ..., n.$$

If A is diagonally dominant then the Jacobi method converges.

93.37. Prove this claim.

93.38. Derive an algorithm that uses the Jacobi method to solve a tridiagonal system. Use as few operations and as little storage as possible.

93.39. Devise an algorithm to estimate the error of the solution of a linear system using single and double precision as suggested. Repeat the example using a tridiagonal solver and your algorithm to estimate the error.

93.40. Show that the sequences $\{x^k\}$ and $\{d^k\}$ generated by the conjugate gradient method (93.16)-(93.17), with $x^1 = 0$ and $d^1 = b$, satisfies for $k = 1, 2, ...,$ (a) $x^k \in K_k(A) = \{b, ..., A^{k-1}b\}$, (b) d^{k+1} is orthogonal to $K_k(A)$, (c) x^k is the projection of x onto $K_k(A)$ with respect to the scalar product $< y, z >= (Ay, z)$.

93.41. The Cbebyshev polynomial $q_k(x)$ if degree k is defined for $-1 \leq x \leq 1$ by the formula $q_k(x) = \cos(k \arccos(x))$. Show that $q_k'(0) \approx k^2$. Deduce from this result that the number of iterations in the conjugate gradient method scales like $\sqrt{\kappa A)}$.

93.42. Compare the GMRES-algorithm for $Ax = b$ with the conjugate gradient method fro the normal equations $A^\top A = A^\top b$.

93.43. The formula $AQ_k = Q_{k+1}H_k$, with H_k an upper Hessenberg matrix ($h_{ij} = 0$ for all $i > j + 1$), defines a recurrence relation for the column vector q_{k+1} of Q_{k+1} in terms of itself and the previous Krylov vectors. (a) Derive this recurrence relation. (b) Implement an algorithm that computes Q_{k+1} and H_k, given a matrix A (this is the *Arnoldi iteration*).

93.44. Prove that $Q_{k+1}^T b = \|b\| e_1$.

93.45. Implement the GMRES-method.

118
Piecewise Linear Interpolation

118.1 Defining the Interpolant

To estimate finite element discretization errors (in time and space), we are led to estimate the interpolation error between a given function $u(x)$ defined on a domain Ω and its *piecewise linear interpolant* u_h taking on the same values as $u(x)$ at the nodes of a triangulation. Here x represents time or a space coordinate. To estimate the interpolation error over the domain Ω it is sufficient to consider the error over each finite element separately, because the interpolant is uniquely defined by the nodal values for each element (interval, triangle or tetrahedron).

Recall that Midpoint Euler (or The Trapezoidal Method) constructs a solution to $\dot{u}(t) = f(u(t))$ for $t > 0$ as a continuous piecewise linear function $u(t)$ of time t.

118.2 To Do 1d

Consider a differentiable function $u(x)$ defined on the interval $[0, h]$ and let u_h be a linear function interpolating $u(x)$ at the end points, that is $u_h(0) = u(0)$ and $u_h(h) = u(h)$. We seek to estimate the interpolation error

$$e_h = u(x) - u_h(x) \quad \text{for } x \in [0, h]. \tag{118.1}$$

Step 1: Reduce to the case $u(0) = u(h) = 0$, by changing $u(x)$ and $u_h(x)$ by the same linear function. Notice that in this case $u_h(x) \equiv 0$ for $x \in [0, h]$.

Step 2: Assume that $u_h(x) \neq u(x)$ for some $x \in (0,h)$. Motivate that $u(x)$ takes on a maximum or minimimum value at some point $\xi \in (0,h)$ and show that $u'(\xi) = 0$.

Step 3: Use the differentiability of $u(x)$ to show that for $x \in [0,h]$

$$\begin{aligned} |u(x) - u(\xi)| = |u(x) - u(\xi) - u'(\xi)(x-\xi)| \leq C|x-\xi|^2 \leq Ch^2, \\ |u'(x) - u'(\xi)| \leq C|x-\xi| \leq Ch. \end{aligned} \tag{118.2}$$

Alternatively, use Taylor's formula or Taylor series with expansion around $x = \xi$.

Step 4: Conclude that

$$\begin{aligned} |u(x) - u_h(x)| &\leq C_0 C h^2, \quad (C_0 \approx \frac{1}{8}) \\ |u'(x) - u_h'(x)| &\leq C_1 C h, \quad (C_1 \approx \frac{1}{2}) \end{aligned} \tag{118.3}$$

where C bounds $|u''(x)|$ for $x \in [0,h]$.

Alternative proof: Let $x \in (0,h)$ and consider the function $g(y)$ defined for $y \in [0,h]$ by

$$g(y) = u(y) - u(0)\frac{h-y}{h} - u(h)\frac{y}{h} - \gamma(x)y(h-y), \tag{118.4}$$

where $\gamma(x)$ is so chosen that $g(x) = 0$. Notice that $g(0) = g(x) = g(h) = 0$ and use the mean-value theorem (first for $g(y)$ twice and then for $g'(y)$ once) to show that $g''(\xi) = 0$ for some $\xi \in (0,h)$ and thus that $\gamma(x) = -\frac{1}{2}u''(\xi)$. Then show that $C_0 = \frac{1}{8}$

118.3 Direct Computation of Interpolation errors

Let $\bar{u}_h$ be a piecewise quadratic interpolant of $u(x)$ interpolating at the endpoints and midpoint of each element. Use

$$\max_{x\in[0,h]} |\bar{u}_h(x) - u_h(x)|, \quad \max_{x\in[0,h]} |\bar{u}_h'(x) - u_h'(x)| \tag{118.5}$$

as direct quantitative estimates of the interpolation errors

$$\max_{x\in[0,h]} |u(x) - u_h(x)|, \quad \max_{x\in[0,h]} |u'(x) - u_h'(x)|. \tag{118.6}$$

Use this technique for estimation of errors in piecewise linear interpolation of different functions.

118.4 To Do in 2d and 3d

Extend to 2d and 3d.

118.5 Compare

- Piecewise Polynomials 1d
- Piecewise Polynomials 2d and 3d

118.6 Piecewise Constant Approximation

In *piecewise constant approximation* the interpolant u_h is defined as a constant on each finite element, e.g, as the mean-value over the element. Recall from Time Stepping Error Analysis

$$\max_{x\in[0,h]} |u(x) - u_h(x)| \leq h \max_{x\in[0,h]} |u'(x)|. \tag{118.7}$$

118.7 L_2-projection onto Piecewise Constants

Define $u_h(x)$ on $[0,h]$ as the mean-value of $u(x)$, that is,

$$u_h(x) = \frac{1}{h}\int_0^h u(y)\,dy \quad x \in [0,h]. \tag{118.8}$$

Show that $u_h(x)$ can be defined as the constant Pu defined by the orthogonality relation

$$\int_0^h (u(y) - Pu)v(y)\,dy = 0 \tag{118.9}$$

for all constant functions $v(y)$ on $[0,h]$. Show that the constant Pu is a best approximation of $u(y)$ in the sense that

$$\int_0^h (u(y) - Pu)^2\,dy \leq \int_0^h (u(y) - v(y))^2\,dy \tag{118.10}$$

for all constant functions $v(y)$. Hint: Write

$$\begin{aligned}\int_0^h (u - Pu)^2\,dy &= \int_0^h (u - Pu)(u - Pu)\,dy + \int_0^h (u - Pu)(Pu - v)\,dy \\ &= \int_0^h (u - Pu)(u - v)\,dy\end{aligned} \tag{118.11}$$

and use Cauchy's inequality for integrals.

Extend to piecewise constant approximation on a partition of an interval. Extend to $2d$ and $3d$.

Note: This is the basic step in the basic error analysis of the finite element method.

119
Quadrature

119.1 Quadrature by Piecewise Polynomial Interpolation

An integral $\int_I u(x)\,dx$ of a function $u(x)$ over an interval I, van be computed by replacing (interpolating) $u(x)$ by a piecewise polynomial interpolant (constant, linear, quadratic,...) u_h on some partition of I into subintervals, and computing the integral $\int_I u_h(x)\,dx$ analytically (as a sum of analytically computable integrals over subintervals). This is called (numerical) *quadrature*. The quadrature is then exact if $u(x)$ is a piecewise polynomial in question.

119.2 Trapezoidal Rule by Linear Approximation

Let $[0,h]$ be an interval and consider the quadrature formula (Trapezoidal Rule):

$$\int_0^h u(x)\,dx \approx \frac{h}{2}(u(0)+u(h)) \tag{119.1}$$

obtained by replacing the given function $u(x)$ by its linear interpolant

$$u_h = (1-\frac{x}{h})u(0) + \frac{x}{h}u(h) \tag{119.2}$$

and computing the integral of u_h analytically.

The quadrature error can be estimated by reducing to the case $u(0) = u(h) = 0$ and then assuming that $u(x)$ is quadratic (linear plus one) so that $u(x) = x(h - x)$. We compute

$$\int_0^h x(h-x)\,dx = \frac{h^3}{6}, \tag{119.3}$$

suggesting the quadrature error estimate (since the second derivative of $x(h - x)$ equals -2):

$$|\int_0^h u(x)\,dx - \frac{h}{2}(u(0) + u(h))| \leq \frac{h^3}{12}\max_{[0,h]}|u''| \tag{119.4}$$

The accurcay of the Trapezoidal rule is thus of order h^2, when normalizing for the length h of the interval, so that over an interval I of unit length partitioned into subintervals of length h, the quadrature error is bounded by $\frac{h^2}{12}\max_I |u''|$.

119.3 To Do

- Estimate the quadrature error in the rectangle and trapezoidal quadrature rules corresponding to piecewise constant and linear linear approximation.
- Compare with Forward/Backward/Midpoint Euler time stepping.
- Contemplate adaptive quadrature with variable subinterval length.

119.4 To Read

- Adaptive Quadrature

158

Linearization and Stability of Initial Value Problems

> The logos of somewome to that base anything, when most characteristically mantissa minus, comes to nullum in the endth: orso, here is nowet badder than the sin of Aha with his cosin Lil, verswaysed on coversvised, and all that's consecants and cotangincies... (Finnegans Wake, James Joyce)

158.1 Introduction

We continue the study of the general initial value problem (210.1), now focussing on the *stability* of solutions, which is a measure of the *sensitivity of solutions to perturbations in given data*. This is a fundamentally important aspect of the behavior of solutions, which we touched upon in Chapter *The general initial value problem*, and which we now consider more closely.

We consider an autonomous problem of the form

$$\dot{u}(t) = f(u(t)) \quad \text{for } 0 < t \leq T,\ u(0) = u^0, \tag{158.1}$$

where $f : \mathbb{R}^d \to \mathbb{R}^d$ is a given bounded Lipschitz continuous function, $u^0 \in \mathbb{R}^d$ is a given initial value, and we seek a solution $u : [0,T] \to \mathbb{R}^d$, where we think of $[0,T]$ as a given time interval. To study the stability of a given solution $u(t)$ to small perturbations in given data, e.g. in the given initial data u^0, we will consider an associated *linearized problem* that arises upon linearizing the function $v \to f(v)$ around the solution $u(t)$.

158.2 Stationary Solutions

We consider first the simplest case of a *stationary solution* $u(t) = \bar{u}$ for $0 \leq t \leq T$, that is a solution $u(t)$ of (158.1) that is independent of time t. Since $\dot{u}(t) = 0$ if $u(t)$ is independent of time, $u(t) = \bar{u}$ is a stationary solution if $f(\bar{u}) = 0$ and $u^0 = \bar{u}$, where $\bar{u} = (\bar{u}_1, ..., \bar{u}_d) \in \mathbb{R}^d$. The equation $f(\bar{u}) = 0$ corresponds to a system of d equations $f_i(\bar{u}_1, ..., \bar{u}_d) = 0$, $i = 1, ..., d$, in d unknowns $\bar{u}_1, ..., \bar{u}_d$, where the f_i are the components of f. We studied such systems of equations in Chapter *Vector-valued functions of several real variables*. Here, we assume the existence of a stationary solution $u(t) = \bar{u}$ so that $\bar{u} \in \mathbb{R}^d$ satisfies the equation $f(\bar{u}) = 0$. In general, there may be several roots $\bar{u}$ of the equation $f(v) = 0$ and thus there may be several stationary solutions. We also refer to a stationary solution $u(t) = \bar{u}$ as an *equilibrium solution*.

EXAMPLE 158.1. The stationary solutions $\bar{u}$ of the Crash model

$$\begin{cases} \dot{u}_1 + \nu u_1 - \kappa u_1 u_2 = \nu & t > 0, \\ \dot{u}_2 + 2\nu u_2 - \nu u_2 u_1 = 0 & t > 0, \end{cases} \tag{158.2}$$

of the form $\dot{u} = f(u)$ with $f(u) = (-\nu u_1 + \kappa u_1 u_2 + \nu, -2\nu u_2 + \nu u_2 u_1)$, are $\bar{u} = (1, 0)$ and $\bar{u} = (2, \frac{\nu}{\kappa})$.

158.3 Linearization at a Stationary Solution

We shall now study perturbations of a given stationary solution under small perturbations of initial data. We thus assume $f(\bar{u}) = 0$ and denote the corresponding equilibrium solution by $\bar{u}(t)$ for $t > 0$, that is $\bar{u}(t) = \bar{u}$ for $t > 0$. We consider the initial value problem (158.1) with $u^0 = \bar{u} + \varphi^0$, where $\varphi^0 \in \mathbb{R}^d$ is a given small perturbation of the initial data $\bar{u}$. We denote the corresponding solution by $u(t)$ and focus attention on the corresponding perturbation in the solution, that is $\psi(t) = u(t) - \bar{u}(t) = u(t) - \bar{u}$. We want to derive a differential equation for the perturbation $\psi(t)$, and to this end we linearize f at $\bar{u}$ and write

$$f(u(t)) = f(\bar{u} + \psi(t)) = f(\bar{u}) + f'(\bar{u})\psi(t) + e(t),$$

where $f'(\bar{u})$ is the Jacobian of $f : \mathbb{R}^d \to \mathbb{R}^d$ at $\bar{u}$ and the error term $e(t)$ is quadratic in $\psi(t)$ (and thus is very small if $\psi(t)$ is small). Since $f(\bar{u}) = 0$ and $u(t)$ satisfies (158.1), we have

$$\dot{\psi}(t) = \frac{d}{dt}(\bar{u} + \psi(t))) = f(u(t)) = f'(\bar{u})\psi(t) + e(t).$$

Neglecting the quadratic term $e(t)$, we are led to a linear initial value problem,

$$\dot{\varphi}(t) = f'(\bar{u})\varphi(t) \quad \text{for } t > 0, \quad \varphi(0) = \varphi^0, \tag{158.3}$$

where $\varphi(t)$ is an approximation of the perturbation $\psi(t) = u(t) - \bar{u}$ up to a second order term. We refer to (158.3) as the *linearized problem* associated to the stationary solution $\bar{u}$ of (158.1). Since $f'(\bar{u})$ is a constant $d \times d$ matrix, we can express the solution to (158.3) using the matrix exponential as

$$\varphi(t) = \exp(tA)\varphi^0 \quad \text{for } 0 < t \leq T, \tag{158.4}$$

where $A = f'(\bar{u})$. We thus have a formula that describes the evolution of perturbation $\varphi(t)$ starting from an initial perturbation $\varphi(0) = \varphi^0$. Depending on the nature of the matrix $\exp(tA)$, the perturbation may increase or decrease with time, reflecting a stronger or lesser sensitivity of the solution $u(t)$ to perturbations in initial data and therefore different stability features of the given problem.

We know that if A is diagonalizable, so that $A = B\Lambda B^{-1}$ where B is a non-singular $d \times d$ matrix and Λ is a diagonal matrix with the eigenvalues $\lambda_1, ..., \lambda_d$ of A on the diagonal, then

$$\varphi(t) = B\exp(t\Lambda)B^{-1}\varphi^0 \quad \text{for } t \geq 0. \tag{158.5}$$

We see that each component of $\varphi(t)$ is a linear combination of $\exp(t\lambda_1)$,..., $\exp(t\lambda_d)$ and the sign of the real part $\text{Re}\,\lambda_i$ of λ_i determines if the corresponding term grows or decays exponentially. If some $\text{Re}\,\lambda_i > 0$, then we have exponential growth of certain perturbations, which indicates that the corresponding stationary solution $\bar{u}$ is *unstable*. On the other hand, if all $\text{Re}\,\lambda_i \leq 0$, then we would expect $\bar{u}$ to be *stable*.

These considerations are qualitative in nature, and to be more precise we should base judgements of stability or instability on quantitative estimates of perturbation growth. In the diagonalizable case, (158.5) implies in the Euclidean vector and matrix norms that

$$\|\varphi(t)\| \leq \|B\|\|B^{-1}\| \max_{i=1,...,d} \exp(t\lambda_i)\|\varphi^0\|. \tag{158.6}$$

We see that the maximal perturbation growth is governed by the maximal exponential factors $\exp(t\lambda_i)$ as well as the factors $\|B\|$ and $\|B^{-1}\|$ related to the transformation matrix B. If the transformation matrix B is orthogonal, then $\|B\| = \|B^{-1}\| = 1$, and the perturbation growth is governed solely by the exponential factors $\exp(t\lambda_i)$. We give this case special attention:

158.4 Stability Analysis when $f'(\bar{u})$ Is Symmetric

If $A = f'(\bar{u})$ is symmetric so that $A = Q\Lambda Q^{-1}$ with Q orthogonal and Λ a diagonal matrix with real diagonal elements λ_i, then

$$\|\varphi(t)\| \leq \max_{i=1,...,d} \exp(t\lambda_i)\|\varphi^0\|. \tag{158.7}$$

In particular, if all eigenvalues $\lambda_i \leq 0$ then perturbations $\varphi(t)$ cannot grow with time, and we say that the solution $\bar{u}$ is stable. On the other hand, if some eigenvalue $\lambda_i > 0$ and the corresponding eigenvector is g_i then $\varphi(t) = \exp(t\lambda_i)g_i$ solves the linearized initial value problem (158.3) with $\varphi^0 = g_i$, and evidently the particular perturbation $\varphi(t)$ grows exponentially. We then say that the solution $\bar{u}$ is *unstable*. Of course, the size of the positive eigenvalues influence the perturbation growth, so that if $\lambda_i > 0$ is small, then then growth is slow and the instability is mild. Likewise, if λ_i is small negative, then the exponential decay is slow.

158.5 Stability Factors

We may express the stability features of a particular perturbation φ^0 through a *stability factor* $S(T, \varphi_0)$ defined as follows:

$$S(T, \varphi_0) = \max_{0 \leq t \leq T} \frac{\|\varphi(t)\|}{\|\varphi^0\|}.$$

where $\varphi(t)$ solves the linearized problem (158.3) with initial data φ^0. The stability factor $S(T, \varphi_0)$ measures the maximal growth of the norm of $\varphi(t)$ over the time interval $[0, T]$ versus the norm of the initial value φ_0.

We can now seek to capture the overall stability features of a stationary solution $\bar{u}$ by maximization over all different perturbations:

$$S(T) = \max_{\varphi^0 \neq 0} S(T, \varphi_0).$$

If the stability factor $S(T)$ is large, then some perturbations grow very much over the time interval $[0, T]$, which indicates a strong sensitivity to perturbations or *instability*. On the other hand, if $S(T)$ is of moderate size then the perturbation growth is moderate, which signifies *stability*. Using the Euclidean matrix norm, we can also express $S(T)$ as

$$S(T) = \max_{0 \leq t \leq T} \|\exp(tA)\|.$$

EXAMPLE 158.2. If $A = f'(\bar{u})$ is symmetric with eigenvalues $\lambda_1, ..., \lambda_d$, then

$$S(T) = \max_{i=1,...,d} \max_{0 \leq t \leq T} \exp(t\lambda_i).$$

In particular, if all $\lambda_i \leq 0$, then $S(T) = 1$.

EXAMPLE 158.3.

The initial value problem for a pendulum takes the form

$$\begin{aligned} \dot{u}_1 = u_2, \quad \dot{u}_2 = -\sin(u_1) \quad \text{for } t > 0, \\ u_1(0) = u_{01},\ u_2(0) = u_{02}, \end{aligned}$$

corresponding to $f(u) = (u_2, -\sin(u_1)$ and the equilibrium solutions are $\bar{u} = (0,0)$ and $\bar{u} = (\pi, 0)$. We have

$$f'(\bar{u}) = \begin{pmatrix} 0 & 1 \\ -\cos(\bar{u}_1) & 0 \end{pmatrix},$$

and the linearized problem at $\bar{u} = (0,0)$ thus takes the form

$$\dot{\varphi}(t) = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \varphi(t) \equiv A_0\varphi(t) \quad \text{for } t > 0, \quad \varphi(0) = \varphi^0,$$

with solution

$$\varphi_1(t) = \varphi_1^0 \cos(t) + \varphi_2^0 \sin(t), \quad \varphi_2(t) = -\varphi_1^0 \sin(t) + \varphi_2^0 \cos(t).$$

It follows by a direct computation (or using that $\begin{pmatrix} \cos(t) & \sin(t) \\ -\sin(t) & \cos(t) \end{pmatrix}$ is an orthogonal matrix), that for $t > 0$

$$\|\varphi(t)\|^2 = \|\varphi_0\|^2,$$

and thus the norm $\|\varphi(t)\|$ of a solution $\varphi(t)$ of the linearized equations is constant in time, which means that the stability factor $S(T) = 1$ for all $T > 0$. We conclude that if the norm of a perturbation is small initially, it will stay small for all time. This means that the equilibrium solution $\bar{u} = (0,0)$ is *stable*. More precisely, if the pendulum is perturbed initially a little from its bottom position, the pendulum will oscillate back and forth around the bottom position with constant amplitude. This fits our direct experimental experience of course.

Note that the linearized operator A_0 is non-symmetric; the eigenvalues of A_0 are purely imaginary $\pm i$, which says that $\|\varphi(t)\| = \|\varphi_0\|$, that is a perturbation neither grows nor decays. Another way to derive this fact is to use the fact that A_0 is *antisymmetric*, that is $A_0^\top = -A_0$, which shows that $(A_0\varphi, \varphi) = (\varphi, A_0^\top \varphi) = -(\varphi, A_0\varphi) = -(A_0\varphi, \varphi)$, and thus $(A_0\varphi, \varphi) = 0$, where $(\cdot,\cdot)$ is the $\mathbb{R}^2$ scalar product. It follows from the equation $\dot{\varphi} = A_0\varphi$ upon multiplication by φ that $0 = (\dot{\varphi}, \varphi) = \frac{1}{2}\frac{d}{dt}(\varphi, \varphi) = \frac{1}{2}\frac{d}{dt}\|\varphi\|^2$, which proves that $\|\varphi(t)\|^2 = \|\varphi_0\|^2$.

The linearized problem at $\bar{u} = (\pi, 0)$ reads

$$\dot{\varphi}(t) = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \varphi(t) \equiv A_\pi\varphi(t) \quad \text{for } t > 0, \quad \varphi(0) = \varphi^0,$$

with symmetric matrix A_π with eigenvalues ± 1. Since one eigenvalue is positive, the stationary solution $\bar{u} = (\pi, 0)$ is unstable. More precisely, the solution is given by

$$\varphi_1 = \frac{\varphi_1^0}{2}(e^t + e^{-t}) + \frac{\varphi_2^0}{2}(e^t - e^{-t}), \quad \varphi_2 = \frac{\varphi_1^0}{2}(e^t - e^{-t}) + \frac{\varphi_2^0}{2}(e^t + e^{-t}),$$

and due to the exponential factor e^t, perturbations will grow exponentially in time, and thus an initially small perturbation will become large as soon as $t \geq 10$ say. Physically, this means that if the pendulum is perturbed initially a little from its top position, the pendulum will eventually move away from the top position, even if the initial perturbation is very small. This fact of course has direct experimental evidence: to balance a pendulum with the weight in the top position is tricky business. Small perturbations quickly grow to large perturbations and the equilibrium solution $(\pi, 0)$ of the pendulum is unstable.

EXAMPLE 158.4. The linearization of the Crash model (158.2) at the equilibrium solution $\bar{u} = (1, 0)$, takes the form

$$\dot{\varphi}(t) = \begin{pmatrix} -\nu & \kappa \\ 0 & -\nu \end{pmatrix} \varphi(t) \equiv A_{\nu,\kappa}\varphi(t) \quad \text{for } t > 0, \quad \varphi(0) = \varphi^0, \quad (158.8)$$

The solution is given by $\varphi_2(t) = \varphi_2^0 \exp(-\nu t)$, and $\varphi_1(t) = t\kappa \exp(-\nu t)\varphi_2^0 + \exp(-\nu t)\varphi_1^0$. Clearly, $\varphi_2(t)$ decays monotonically to zero and so does $\varphi_1(t)$ if $\kappa = 0$. But, if $\kappa \neq 0$ then $\varphi_1(t)$ reaches the following value, assuming for simplicity that $\varphi_{01} = 0$,

$$\varphi_1(\nu^{-1}) = \nu^{-1}\kappa \exp(-1)\varphi_2^0,$$

which contains the factor ν^{-1} that is large if ν is small. In other words, the stability factor $S(\nu^{-1}) \sim \nu^{-1}$, which is large if ν is small. Eventually, however, $\varphi_1(t)$ decays to zero. As a result, the equilibrium solution $(1, 0)$ is stable only to small perturbations, since we saw in the Chapter The Crash model that $(1, 0)$ is unstable to perturbations above a certain threshold depending on λ. Note that here the Jacobian $f'(\bar{u}) = A_{\nu,\kappa}$ has a double eigenvalue $-\nu$, but $A_{\nu,\kappa}$ is non-symmetric and the space of eigenvectors is one-dimensional and is spanned by $(1, 0)$. As a result, the term $t\kappa \exp(-\nu t)\varphi_2^0$ with linear growth in t appears; thus in this highly non-symmetric problem (if ν is small), large perturbation growth $\sim \nu^{-1}$ is possible although all eigenvalues are non-positive.

The matrix $A_{\nu,\kappa}$ is an example of a *non-normal* matrix. A non-normal matrix A is a matrix such that $A^\top A \neq AA^\top$. A non-normal matrix may or may not be diagonalizable, and if diagonalizable so that $A = B\Lambda B^{-1}$, we may have $\|B\|$ or $\|B^{-1}$ large, resulting in large stability factors in the corresponding linearized problem, as we just saw (cf. Problem 158.5).

The linearization at the equilibrium solution $\bar{u} = (2, \frac{\nu}{\kappa})$ takes the form

$$\dot{\varphi}(t) = \begin{pmatrix} 0 & 2\kappa \\ \frac{\nu^2}{\kappa} & 0 \end{pmatrix} \varphi(t) \quad \text{for } t > 0, \quad \varphi(0) = \varphi^0. \quad (158.9)$$

The eigenvalues of the Jacobian are $\pm\sqrt{2}\nu$ and the solution is a linear combination of $\exp(\sqrt{2}\nu t)$ and $\exp(-\sqrt{2}\nu t)$ and thus has one exponentially growing part with growth factor $\exp(\sqrt{2}\nu t)$. The equilibrium solution $u = (2, \frac{\nu}{\kappa})$ is thus unstable.

158.6 Stability of Time-Dependent Solutions

We now seek to extend the scope to linearization and linearized stability for a time-dependent solution $\bar{u}(t)$ of (158.1). We want to study solutions of the form $u(t) = \bar{u}(t) + \psi(t)$, where $\psi(t)$ is a perturbation. Using $\frac{d}{dt}\bar{u} = f(\bar{u})$ and linearizing f at $\bar{u}(t)$, we obtain

$$\frac{d}{dt}(\bar{u} + \psi)(t) = f(\bar{u}(t)) + f'(\bar{u}(t))\psi(t) + e(t),$$

with $e(t)$ quadratic in $\psi(t)$. This leads to the linearized equation

$$\dot{\varphi}(t) = A(t)\varphi(t) \quad \text{for } t > 0, \varphi(0) = \varphi_0, \tag{158.10}$$

where $A(t) = f'(\bar{u}(t))$ is an $d \times d$ matrix that now depends on t if $\bar{u}(t)$ depends on t. We have no analytical solution formula to this general problem and thus although the stability properties of the given solution $\bar{u}(t)$ are expressed through the solutions $\varphi(t)$ of the linearized problem (158.10), it may be difficult to analytically assess these properties. We may define stability factors $S(T, \varphi_0)$ and $S(T)$ just as above, and we may say that a solution $\bar{u}(t)$ is stable if $S(T)$ is moderately large, and unstable if $S(T)$ is large. To determine $S(T)$ in general, we have to use numerical methods and solve (158.10) with different initial data φ^0. We return to the computation of stability factors in the next chapter on adaptive solvers for initial value problems.

158.7 Sum Up

The question of stability of solutions to initial value problems is of fundamental importance. We can give an affirmative answer in the case of a stationary solution with corresponding symmetric Jacobian. In this case a positive eigenvalue signifies instability, with the instability increasing with increasing eigenvalue, and all eigenvalues non-positive means stability. The case of an anti-symmetric Jacobian also signifies stability with the norm of perturbations being constant in time. If the Jacobian is non-normal we have to watch out and remember that just looking at the sign of the real part of eigenvalues may be misleading: in the non-normal case algebraic growth may in fact dominate slow exponential decay for finite time. In these cases and also for time-dependent solutions, an analytical stability analysis may be out of reach and the desired information about stability may be obtained by numerical solution of the associated linearized problem.

Chapter 158 Problems

158.1. Determine the stationary solutions to the system

$$\begin{aligned}\dot{u}_1 &= u_2(1-u_1^2),\\ \dot{u}_2 &= 2-u_1u_2,\end{aligned}$$

and study the stability of these solutions.

158.2. Determine the stationary solutions to the following system (Minea's equation) for different values of $\delta > 0$ and γ,

$$\begin{aligned}\dot{u}_1 &= -u_1-\delta(u_2^2+u_3^2)+\gamma,\\ \dot{u}_2 &= -u_2-\delta u_1u_2,\\ \dot{u}_3 &= -u_3-\delta u_1u_3,\end{aligned}$$

and study the stability of these solutions.

158.3. Determine the stationary solutions of the system (158.1) with (a) $f(u) = (u_1(1-u_2), u_2(1-u_1))$, (b) $f(u) = (-2(u_1-10)+u_2\exp(u_1), -2u_2-u_2\exp(u_1))$, (c) $f(u) = (u_1+u_1u_2^2+u_1u_3^2, -u_1+u_2-u_2u_3+u_1u_2u_3, u_2+u_3-u_1^2)$, and study the stability of these solutions.

158.4. Determine the stationary solutions of the system (158.1) with (158.1) with (a) $f(u) = (-1001u_1+999u_2, 999u_1-1001u_2)$, (b) $f(u) = (-u_1+3u_2+5u_3, -4u_2+6u_3, u_3)$, (c) $f(u) = (u_2, -u_1-4u_2)$, and study the stability of these solutions.

158.5. Analyze the stability of the following variant of the linearized problem (158.8) with $\epsilon > 0$ small,

$$\dot{\varphi}(t) = \begin{pmatrix}-\nu & \kappa\\ \epsilon & -\nu\end{pmatrix}\varphi(t) \equiv A_{\nu,\kappa,\epsilon}\varphi(t) \quad \text{for } t>0, \quad \varphi(0)=\varphi^0, \tag{158.11}$$

by diagonalizing the matrix $\equiv A_{\nu,\kappa,\epsilon}$. Note that the diagonalization degenerates as ϵ tends to zero (that is, the two eigenvectors become parallel). Check if $A_{\nu,\kappa,\epsilon}$ is a normal or non-normal matrix.